BA

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 November 2003 (27.11.2003)

PCT

(10) International Publication Number
WO 03/098475 A1

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/US03/13145

(22) International Filing Date: 29 April 2003 (29.04.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/376,651	29 April 2002 (29.04.2002)	US
60/376,652	29 April 2002 (29.04.2002)	US
10/371,464	21 February 2003 (21.02.2003)	US
10/425,685	28 April 2003 (28.04.2003)	US

(71) Applicant: SONY ELECTRONICS, INC. [US/US]; 1 Sony Drive, Park Ridge, NJ 07656 (US).

(72) Inventors: VISHARAM, Mohammed, Zubair; 444 Saratoga Avenue, #27K, Santa Clara, CA 95050 (US). TABATABAI, Ali; 10265 E. Estates Drive, Cupertino, CA 95014 (US). WALKER, Toby; 604 Galer Street, #321, Seattle, WA 98109 (US).

(74) Agents: SALTER, James, H. et al.; Blakely, Sokoloff, Taylor & Zafman, 12400 Wishire Boulevard, 7th Floor, Los Angeles, CA 90025-1026 (US).

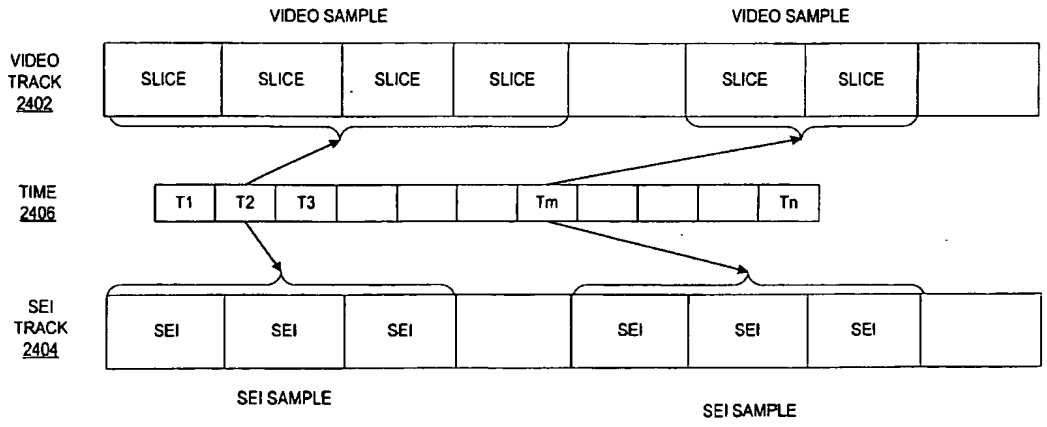
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SUPPORTING ADVANCED CODING FORMATS IN MEDIA FILES



(57) Abstract: One or more descriptions pertaining to multimedia data are identified (figure 1) and included into supplemental enhancement information (SEI) associated with the multimedia data (figure 24). Subsequently, the SEI containing the one or more descriptions is transmitted to a decoding system for optional use in decoding of the multimedia data (figure 2).

WO 03/098475 A1

SUPPORTING ADVANCED CODING FORMATS IN MEDIA FILES

RELATED APPLICATIONS

This application is related to and claims the benefit of U.S. Provisional Patent applications serial numbers 60,376,651 filed April 29, 2002, and 60/376,652 filed April 29, 2002, which are hereby incorporated by reference. This application is also related to U.S. Patent Application serial number 10/371,464 filed February 21, 2003.

FIELD OF THE INVENTION

The invention relates generally to the storage and retrieval of audiovisual content in a multimedia file format and particularly to file formats compatible with the ISO media file format.

COPYRIGHT NOTICE/PERMISSION

BACKGROUND OF THE INVENTION

In the wake of rapidly increasing demand for network, multimedia, database and other digital capacity, many multimedia coding and storage schemes have evolved. One of the well known file formats for encoding and storing audiovisual data is the QuickTime® file format developed by Apple Computer Inc. The QuickTime file format was used as the starting point for creating the International Organization for Standardization (ISO)

Multimedia file format, ISO/IEC 14496-12, Information Technology – Coding of audio-visual objects – Part 12: ISO Media File Format (also known as the ISO file format), which was, in turn, used as a template for two standard file formats: (1) For an MPEG-4 file format developed by the Moving Picture Experts Group, known as MP4 (ISO/IEC 14496-14, Information Technology -- Coding of audio-visual objects -- Part 14: MP4 File Format); and (2) a file format for JPEG 2000 (ISO/IEC 15444-1), developed by Joint Photographic Experts Group (JPEG).

The ISO media file format is composed of object-oriented structures referred to as boxes (also referred to as atoms or objects). The two important top-level boxes contain either media data or metadata. Most boxes describe a hierarchy of metadata providing declarative, structural and temporal information about the actual media data. This collection of boxes is contained in a box known as the movie box. The media data itself may be located in media data boxes or externally. The collective hierarchy of metadata boxes providing information about a particular media data are known as tracks.

The primary metadata is the movie object. The movie box includes track boxes, which describe temporally presented media data. The media data for a track can be of various types (e.g., video data, audio data, binary format screen representations (BIFS), etc.). Each track is further divided into samples (also known as access units or pictures). A sample represents a unit of media data at a particular time point. Sample metadata is contained in a set of sample boxes. Each track box contains a sample table box metadata box, which contains boxes that provide the time for each sample, its size in bytes, and so forth. A sample is the smallest data entity which can represent timing, location, and other metadata information. Samples may be grouped into chunks that include sets of consecutive samples. Chunks can be of different sizes and include samples of different sizes.

Recently, MPEG's video group and Video Coding Experts Group (VCEG) of International Telecommunication Union (ITU) began working together as a Joint Video Team (JVT) to develop a new video coding/decoding (codec) standard referred to as ITU Recommendation H.264 or MPEG-4-Part 10, Advanced Video Codec (AVC) or JVT

codec. These terms, and their abbreviations such as H.264, JVT, and AVC are used interchangeably here.

The JVT codec design distinguished between two different conceptual layers, the Video Coding Layer (VCL), and the Network Abstraction Layer (NAL). The VCL contains the coding related parts of the codec, such as motion compensation, transform coding of coefficients, and entropy coding. The output of the VCL is slices, each of which contains a series of macroblocks and associated header information. The NAL abstracts the VCL from the details of the transport layer used to carry the VCL data. It defines a generic and transport independent representation for information above the level of the slice. The NAL defines the interface between the video codec itself and the outside world. Internally, the NAL uses NAL packets. A NAL packet includes a type field indicating the type of the payload plus a set of bits in the payload. The data within a single slice can be divided further into different data partitions.

In many existing video coding formats, the coded stream data includes various kinds of headers containing parameters that control the decoding process. For example, the MPEG-2 video standard includes sequence headers, enhanced group of pictures (GOP), and picture headers before the video data corresponding to those items. In JVT, the information needed to decode VCL data is grouped into parameter sets. Each parameter set is given an identifier that is subsequently used as a reference from a slice. Instead of sending the parameter sets inside (in-band) the stream, they can be sent outside (out-of-band) the stream.

Existing file formats do not provide a facility for storing the parameter sets associated with coded media data; nor do they provide a means for efficiently linking media data (i.e., samples or sub-samples) to parameters sets so that parameter sets can be efficiently retrieved and transmitted.

In the ISO media file format, the smallest unit that can be accessed without parsing media data is a sample, i.e., a whole picture in AVC. In many coded formats, a sample can be further divided into smaller units called sub-samples (also referred to as sample fragments or access unit fragments). In the case of AVC, a sub-sample corresponds to a

slice. However, existing file formats do not support accessing sub-parts of a sample. For systems that need to flexibly form data stored in a file into packets for streaming, this lack of access to sub-samples hinders flexible packetization of JVT media data for streaming.

Another limitation of existing storage formats has to do with switching between stored streams with different bandwidth in response to changing network conditions when streaming media data. In a typical streaming scenario, one of the key requirements is to scale the bit rate of the compressed data in response to changing network conditions. This is typically achieved by encoding multiple streams with different bandwidth and quality settings for representative network conditions and storing them in one or more files. The server can then switch among these pre-coded streams in response to network conditions. In existing file formats, switching between streams is only possible at samples that do not depend on prior samples for reconstruction. Such samples are referred to as I-frames. No support is currently provided for switching between streams at samples that depend on prior samples for reconstruction (i.e., a P-frame or a B-frame that depend on multiple samples for reference).

The AVC standard provides a tool known as switching pictures (called SI- and SP-pictures) to enable efficient switching between streams, random access, and error resilience, as well as other features. A switching picture is a special type of picture whose reconstructed value is exactly equivalent to the picture it is supposed to switch to. Switching pictures can use reference pictures differing from those used to predict the picture that they match, thus providing more efficient coding than using I-frames. To use switching pictures stored in a file efficiently it is necessary to know which sets of pictures are equivalent and to know which pictures are used for prediction. Existing file formats do not provide this information and therefore this information must be extracted by parsing the coded stream, which is inefficient and slow.

Thus, there is a need to enhance storage methods to address the new capabilities provided by emerging video coding standards and to address the existing limitations of those storage methods.

SUMMARY OF THE INVENTION

One or more descriptions pertaining to multimedia data are identified and included into supplemental enhancement information (SEI) associated with the multimedia data. Subsequently, the SEI containing the descriptions is transmitted to a decoding system for optional use in decoding of the multimedia data.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

Figure 1 is a block diagram of one embodiment of an encoding system;

Figure 2 is a block diagram of one embodiment of a decoding system;

Figure 3 is a block diagram of a computer environment suitable for practicing the invention;

Figure 4 is a flow diagram of a method for storing sub-sample metadata at an encoding system;

Figure 5 is a flow diagram of a method for utilizing sub-sample metadata at a decoding system;

Figure 6 illustrates an extended MP4 media stream model with sub-samples;

Figures 7A – 7K illustrate exemplary data structures for storing sub-sample metadata;

Figure 8 is a flow diagram of a method for storing parameter set metadata at an encoding system;

Figure 9 is a flow diagram of a method for utilizing parameter set metadata at a decoding system;

Figures 10A – 10E illustrate exemplary data structures for storing parameter set metadata;

Figure 11 illustrates an exemplary enhanced group of pictures (GOP);

Figure 12 is a flow diagram of a method for storing sequences metadata at an encoding system;

Figure 13 is a flow diagram of a method for utilizing sequences metadata at a decoding system;

Figures 14A – 14E illustrate exemplary data structures for storing sequences metadata;

Figures 15A and 15B illustrate the use of a switch sample set for bit stream switching;

Figure 15C is a flow diagram of one embodiment of a method for determining a point at which a switch between two bit streams is to be performed;

Figure 16 is a flow diagram of a method for storing switch sample metadata at an encoding system;

Figure 17 is a flow diagram of a method for utilizing switch sample metadata at a decoding system;

Figure 18 illustrates an exemplary data structure for storing switch sample metadata;

Figures 19A and 19B illustrate the use of a switch sample set to facilitate random access entry points into a bit stream;

Figure 19C is a flow diagram of one embodiment of a method for determining a random access point for a sample;

Figures 20A and 20B illustrate the use of a switch sample set to facilitate error recovery;

Figure 20C is a flow diagram of one embodiment of a method for facilitating error recovery when sending a sample;

Figures 21 and 22 illustrate storage of parameter set metadata according to some embodiments of the present invention; and

Figures 23-26 illustrate storage of supplemental enhancement information (SEI) according to some embodiments of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In the following detailed description of embodiments of the invention, reference is made to the accompanying drawings in which like references indicate similar elements, and in which is shown, by way of illustration, specific embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention, and it is to be understood that other embodiments may be utilized and that logical, mechanical, electrical, functional and other changes may be made without departing from the scope of the present invention. The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the present invention is defined only by the appended claims.

Overview

Beginning with an overview of the operation of the invention, **Figure 1** illustrates one embodiment of an encoding system 100. The encoding system 100 includes a media encoder 104, a metadata generator 106 and a file creator 108. The media encoder 104 receives media data that may include video data (e.g., video objects created from a natural source video scene and other external video objects), audio data (e.g., audio objects created from a natural source audio scene and other external audio objects), synthetic objects, or any combination of the above. The media encoder 104 may consist of a number of individual encoders or include sub-encoders to process various types of media data. The media encoder 104 codes the media data and passes it to the metadata generator 106. The metadata generator 106 generates metadata that provides information about the media data according to a media file format. The media file format may be derived from the ISO media file format (or any of its derivatives such as MPEG-4, JPEG 2000, etc.), QuickTime or any other media file format, and also include some additional data structures. In one embodiment, additional data structures are defined to store metadata pertaining to sub-samples within the media data. In another embodiment, additional data structures are defined to store metadata linking portions of media data (e.g., samples or sub-samples) to corresponding parameter sets which include decoding information that has been

traditionally stored in the media data. In yet another embodiment, additional data structures are defined to store metadata pertaining to various groups of samples within the metadata that are created based on inter-dependencies of the samples in the media data. In still another embodiment, an additional data structure is defined to store metadata pertaining to switch sample sets associated with the media data. A switch sample set refers to a set of samples that have identical decoding values but may depend on different samples. In yet other embodiments, various combinations of the additional data structures are defined in the file format being used. These additional data structures and their functionality will be described in greater detail below.

The file creator 108 is responsible for storing the coded media data and the metadata. In one embodiment, the coded media data and the associated metadata (e.g., sub-sample metadata, parameter set metadata, group sample metadata, or switch sample metadata) are stored in the same file. The structure of this file is defined by the media file format.

In another embodiment, all or some types of the metadata are stored separately from the media data. For example, parameter set metadata may be stored separately from the media data. Specifically, the file creator 108 may include a media data file creator 114 to form a file with the coded media data, a metadata file creator 112 to form a file with the metadata, and a synchronizer 116 to synchronize the media data with the corresponding metadata. The storage of the separated metadata and its synchronization with the media data will be discussed in greater detail below.

In one embodiment, the metadata file creator 112 is responsible for storing supplemental enhancement information (SEI) messages associated with the media data as metadata separately from the media data. SEI messages represent optional data for use in the decoding of the media data. It is not necessary for a decoder to use the SEI data because its lack would not hamper the decoding operation. In one embodiment, the SEI messages are used to include descriptions of the media data. The descriptions are defined according to the MPEG-7 standards and consist of descriptors and description schemes. Descriptors represent features of audiovisual content and define the syntax and the

semantics of each feature representation. Examples of descriptors include color descriptors, texture descriptors, motion descriptors, etc. Description schemes (DS) specify the structure and semantics of the relationships between their components. These components may be both descriptors and description schemes. The use of descriptions improves searching and viewing of the media data once it is decoded. Due to the optional nature of the SEI messages, the inclusion of descriptions into the SEI messages does not negatively affect the decoding operations because the decoder does not need to use the SEI messages unless it has the capability and specific configuration that allow such use. The storage of the SEI messages as metadata will be discussed in greater detail below.

The files created by the file creator 108 are available on a channel 110 for storage or transmission.

Figure 2 illustrates one embodiment of a decoding system 200. The decoding system 200 includes a metadata extractor 204, a media data stream processor 206, a media decoder 210, a compositor 212 and a renderer 214. The decoding system 200 may reside on a client device and be used for local playback. Alternatively, the decoding system 200 may be used for streaming data and have a server portion and a client portion communicating with each other over a network (e.g., Internet) 208. The server portion may include the metadata extractor 204 and the media data stream processor 206. The client portion may include the media decoder 210, the compositor 212 and the renderer 214.

The metadata extractor 204 is responsible for extracting metadata from a file stored in a database 216 or received over a network (e.g., from the encoding system 100). The file may or may not include media data associated with the metadata being extracted. The metadata extracted from the file includes one or more of the additional data structures described above.

The extracted metadata is passed to the media data stream processor 206 which also receives the associated coded media data. The media data stream processor 206 uses the metadata to form a media data stream to be sent to the media decoder 210. In one embodiment, the media data stream processor 206 uses metadata pertaining to sub-samples

to locate sub-samples in the media data (e.g., for packetization). In another embodiment, the media data stream processor 206 uses metadata pertaining to parameter sets to link portions of the media data to its corresponding parameter sets. In yet another embodiment, the media data stream processor 206 uses metadata defining various groups of samples within the metadata to access samples in a certain group (e.g., for scalability by dropping a group containing samples on which no other samples depend to lower the transmitted bit rate in response to transmission conditions). In still another embodiment, the media data stream processor 206 uses metadata defining switch sample sets to locate a switch sample that has the same decoding value as the sample it is supposed to switch to but does not depend on the samples on which this resultant sample would depend on (e.g., to allow switching to a stream with a different bit-rate at a P-frame or B-frame).

Once the media data stream is formed, it is sent to the media decoder 210 either directly (e.g., for local playback) or over a network 208 (e.g., for streaming data) for decoding. The compositor 212 receives the output of the media decoder 210 and composes a scene which is then rendered on a user display device by the renderer 214.

The following description of **Figure 3** is intended to provide an overview of computer hardware and other operating components suitable for implementing the invention, but is not intended to limit the applicable environments. **Figure 3** illustrates one embodiment of a computer system suitable for use as a metadata generator 106 and/or a file creator 108 of **Figure 1**, or a metadata extractor 204 and/or a media data stream processor 206 of **Figure 2**.

The computer system 340 includes a processor 350, memory 355 and input/output capability 360 coupled to a system bus 365. The memory 355 is configured to store instructions which, when executed by the processor 350, perform the methods described herein. Input/output 360 also encompasses various types of computer-readable media, including any type of storage device that is accessible by the processor 350. One of skill in the art will immediately recognize that the term "computer-readable medium/media" further encompasses a carrier wave that encodes a data signal. It will also be appreciated that the system 340 is controlled by operating system software executing in memory 355.

Input/output and related media 360 store the computer-executable instructions for the operating system and methods of the present invention. Each of the metadata generator 106, the file creator 108, the metadata extractor 204 and the media data stream processor 206 that are shown in **Figures 1and 2** may be a separate component coupled to the processor 350, or may be embodied in computer-executable instructions executed by the processor 350. In one embodiment, the computer system 340 may be part of, or coupled to, an ISP (Internet Service Provider) through input/output 360 to transmit or receive media data over the Internet. It is readily apparent that the present invention is not limited to Internet access and Internet web-based sites; directly coupled and private networks are also contemplated.

It will be appreciated that the computer system 340 is one example of many possible computer systems that have different architectures. A typical computer system will usually include at least a processor, memory, and a bus coupling the memory to the processor. One of skill in the art will immediately appreciate that the invention can be practiced with other computer system configurations, including multiprocessor systems, minicomputers, mainframe computers, and the like. The invention can also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network.

Sub-Sample Accessibility

Figures 4 and 5 illustrate processes for storing and retrieving sub-sample metadata that are performed by the encoding system 100 and the decoding system 200 respectively. The processes may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both. For software-implemented processes, the description of a flow diagram enables one skilled in the art to develop such programs including instructions to carry out the processes on suitably configured computers (the processor of the computer executing the instructions from computer-readable media, including memory). The computer-executable instructions may be written

in a computer programming language or may be embodied in firmware logic. If written in a programming language conforming to a recognized standard, such instructions can be executed on a variety of hardware platforms and for interface to a variety of operating systems. In addition, the embodiments of the present invention are not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings described herein. Furthermore, it is common in the art to speak of software, in one form or another (e.g., program, procedure, process, application, module, logic...), as taking an action or causing a result. Such expressions are merely a shorthand way of saying that execution of the software by a computer causes the processor of the computer to perform an action or produce a result. It will be appreciated that more or fewer operations may be incorporated into the processes illustrated in **Figures 4 and 5** without departing from the scope of the invention and that no particular order is implied by the arrangement of blocks shown and described herein.

Figure 4 is a flow diagram of one embodiment of a method 400 for creating sub-sample metadata at the encoding system 100. Initially, method 400 begins with processing logic receiving a file with encoded media data (processing block 402). Next, processing logic extracts information that identifies boundaries of sub-samples in the media data (processing block 404). Depending on the file format being used, the smallest unit of the data stream to which a time attribute can be attached is referred to as a sample (as defined by the ISO media file format or QuickTime), an access unit (as defined by MPEG-4), or a picture (as defined by JVT), etc. A sub-sample represents a contiguous portion of a data stream below the level of a sample.. The definition of a sub-sample depends on the coding format but, in general, a sub-sample is a meaningful sub-unit of a sample that may be decoded as a singly entity or as a combination of sub-units to obtain a partial reconstruction of a sample. A sub-sample may also be called an access unit fragment. Often, sub-samples represent divisions of a sample's data stream so that each sub-sample has few or no dependencies on other sub-samples in the same sample. For example, in

JVT, a sub-sample is a NAL packet. Similarly, for MPEG-4 video, a sub-sample would be a video packet.

In one embodiment, the encoding system 100 operates at the Network Abstraction Layer defined by JVT as described above. The JVT media data stream consists of a series of NAL packets where each NAL packet (also referred to as a NAL unit) contains a header part and a payload part. One type of NAL packet is used to include coded VCL data for each slice, or a single data partition of a slice. In addition, a NAL packet may be an information packet including SEI messages. In JVT, a sub-sample could be a complete NAL packet with both header and payload.

At processing block 406, processing logic creates sub-sample metadata that defines sub-samples in the media data. In one embodiment, the sub-sample metadata is organized into a set of predefined data structures (e.g., a set of boxes). The set of predefined data structures may include a data structure containing information about the size of each sub-sample, a data structure containing information about the total number of sub-samples in each sample, a data structure containing information describing each sub-sample (e.g., what is defined as a sub-sample), a data structure containing information about the total number of sub-samples in each chunk, a data structure containing information about the priority of each sub-sample, or any other data structures containing data pertaining to the sub-samples.

Next, in one embodiment, processing logic determines whether any data structure contains a repeated sequence of data (decision box 408). If this determination is positive, processing logic converts each repeated sequence of data into a reference to a sequence occurrence and the number of times the repeated sequence occurs (processing block 410).

Afterwards, at processing block 412, processing logic includes the sub-sample metadata into a file associated with media data using a specific media file format (e.g., the JVT file format). Depending on the media file format, the sub-sample metadata may be stored with sample metadata (e.g., sub-sample data structures may be included in a sample table box containing sample data structures) or independently from the sample metadata.

Figure 5 is a flow diagram of one embodiment of a method 500 for utilizing sub-sample metadata at the decoding system 200. Initially, method 500 begins with processing logic receiving a file associated with encoded media data (processing block 502). The file may be received from a database (local or external), the encoding system 100, or from any other device on a network. The file includes sub-sample metadata that defines sub-samples in the media data.

Next, processing logic extracts the sub-sample metadata from the file (processing block 504). As discussed above, the sub-sample metadata may be stored in a set of data structures (e.g., a set of boxes).

Further, at processing block 506, processing logic uses the extracted metadata to identify sub-samples in the encoded media data (stored in the same file or in a different file) and combines various sub-samples into packets to be sent to a media decoder, thus enabling flexible packetization of media data for streaming (e.g., to support error resilience, scalability, etc.).

Exemplary sub-sample metadata structures will now be described with reference to an extended ISO media file format (referred to as an extended MP4). It will be obvious to one versed in the art that other media file formats could be easily extended to incorporate similar data structures for storing sub-sample metadata.

Figure 6 illustrates the extended MP4 media stream model with sub-samples. Presentation data (e.g., a presentation containing synchronized audio and video) is represented by a movie 602. The movie 602 includes a set of tracks 604. Each track 604 refers to a media data stream. Each media data stream is divided into samples 606. Each sample 606 represents a unit of media data at a particular time point. A sample 606 is further divided into sub-samples 608. In the JVT standard, a sub-sample 608 may represent a NAL packet or unit, such as a single slice of a picture, one data partition of a slice with multiple data partitions, an in-band parameter set, or an SEI information packet. Alternatively, a sub-sample 606 may represent any other structured element of a sample, such as the coded data representing a spatial or temporal region in the media. In one

embodiment, any partition of the coded media data according to some structural or semantic criterion can be treated as a sub-sample.

A track extends box is used to identify samples in the track fragments when movie fragments are used to provide information on each sample's duration and size, specify each sample's degradation priority, and other sample information. A degradation priority defines the importance of a sample, i.e., it defines how the sample's absence (e.g., due to its loss during transmission) can affect the quality of the movie. In one embodiment, the track extends box is extended to include the default information on sub-samples within the track fragment boxes. This information may include, for example, sub-sample sizes and references to sub-sample descriptions.

A track may be divided into fragments. Each fragment can contain zero or more contiguous runs of samples. A track fragment run box identifies samples in the track fragment, provides information on duration and size of each sample in the track fragment, and other information pertaining to the samples stored in the track fragment. A track fragment header box identifies default data values that are used in the track fragment run box. In one embodiment, the track fragment run box and track fragment header box are extended to include information on sub-samples within the track fragment. The extended information in the track fragment run box may include, for example, the number of sub-samples in each sample stored in the track fragment, each sub-sample's size, references to sub-sample descriptions, and a set of flags. The set of flags indicate whether the track fragment stores media data in chunks of samples or sub-samples, whether sub-sample data is present in the track fragment run box, and whether each sub-sample has size data and/or description reference data present in the track fragment run box. The extended information in the track fragment header box may include, for example, default values of flags indicating whether each sub-sample has size data and/or description reference data present.

Figures 7A – 7L illustrate exemplary data structures for storing sub-sample metadata.

Referring to **Figure 7A**, a sample table box 700 that contains sample metadata boxes defined by the ISO Media File Format is extended to include sub-sample access

boxes such as a sub-sample size box 702, a sub-sample description association box 704, a sub-sample to sample box 706 and a sub-sample description box 708. In one embodiment, the sub-sample access boxes also include a sub-sample to chunk box and a priority box. In one embodiment, the use of sub-sample access boxes is optional.

Referring to **Figure 7B**, a sample 710 may be, for example, divisible into slices such as a slice 712, data partitions such as partitions 714 and regions of interest (ROIs) such as a ROI 716. Each of these examples represents a different kind of division of samples into sub-samples. Sub-samples within a single sample may have different sizes.

A sub-sample size box 718 contains a version field that specifies the version of the sub-sample size box 718, a sub-sample size field specifying the default sub-sample size, a sub-sample count field to provide the number of sub-samples in the track, and an entry size field specifying the size of each sub-sample. If the sub-sample size field is set to 0, then the sub-samples have different sizes that are stored in the sub-sample size table 720. If the sub-sample size field is not set to 0, it specifies the constant sub-sample size, indicating that the sub-sample size table 720 is empty. The table 720 may have a fixed size of 32-bit or variable length field for representing the sub-sample sizes. If the field is varying length, the sub-sample table contains a field that indicates the length in bytes of the sub-sample size field.

Referring to **Figure 7C**, a sub-sample to sample box 722 includes a version field that specifies the version of the sub-sample to sample box 722, an entry count field that provides the number of entries in the table 723. Each entry in the sub-sample to sample table contains a first sample field that provides the index of the first sample in the run of samples sharing the same number of sub-samples-per-sample, and a sub-samples per sample field that provides the number of sub-samples in each sample within a run of samples.

The table 723 can be used to find the total number of sub-samples in the track by computing how many samples are in a run, multiplying this number by the appropriate sub-samples-per-sample, and adding the results of all the runs together.

In other embodiments, sub-samples may be grouped as chunks, rather than samples. Then, a sub-sample to chunk box is used to identify sub-samples within a chunk. The sub-sample to chunk box stores information on the index of the first chunk in the run of chunks sharing the same number of sub-samples, the number of sub-samples in each chunk and the index for the sub-sample description. The sub-sample to chunk box can be used to find a chunk that contains a specific sub-sample, the position of the sub-sample in the chunk and the description of this sub-sample. In one embodiment, when sub-samples are grouped as chunks, the sub-sample to sample box 722 is not present. Similarly, when sub-samples are grouped as samples, the sub-sample to chunk box is not present.

As discussed above, the sub-sample access boxes may include a priority box that specifies the degradation priority for each sub-sample. A degradation priority defines the importance of a sub-sample, i.e., it defines how the sub-sample's absence (e.g., due to its loss during transmission) can affect the quality of the decoded media data. The size of the priority box will be defined by the number of sub-samples in the track, as can determined from the sub-sample to sample box 722 or the sub-sample to chunk box.

Referring to **Figure 7D**, a sub-sample description association box 724 includes a version field that specifies the version of the sub-sample description association box 724, a description type identifier that indicates the type of sub-samples being described (e.g., NAL packets, regions of interest, etc.), and an entry count field that provides the number of entries in the table 726. Each entry in table 726 includes a sub-sample description type identifier field indicating a sub-sample description ID and a first sub-sample field that gives the index of the first sub-sample in a run of sub-samples which share the same sub-sample description ID.

The sub-sample description type identifier controls the use of the sub-sample description ID field. That is, depending on the type specified in the description type identifier, the sub-sample description ID field may itself specify a description ID that directly encodes the sub-samples descriptions inside the ID itself or the sub-sample description ID field may serve as an index to a different table (i.e., a sub-sample description table described below)? For example, if the description type identifier

indicates a JVT description, the sub-sample description ID field may include a code specifying the characteristics of JVT sub-samples. In this case, the sub-sample description ID field may be a 32-bit field, with the least significant 8 bits used as a bit-mask to represent the presence of predefined data partition inside a sub-sample and the higher order 24 bits used to represent the NAL packet type or for future extensions.

Referring to **Figure 7E**, a sub-sample description box 728 includes a version field that specifies the version of the sub-sample description box 728, an entry count field that provides the number of entries in the table 730, a description type identifier field that provides a description type of a sub-sample description field providing information about the characteristics of the sub-samples, and a table containing one or more sub-sample description entries 730. The sub-sample description type identifies the type to which the descriptive information relates and corresponds to the same field in the sub-sample description association table 724. Each entry in table 730 contains a sub-sample description entry with information about the characteristics of the sub-samples associated with this description entry. The information and format of the description entry depend on the description type field. For example, when the description type is parameter set, then each description entry will contain the value of the parameter set.

The descriptive information may relate to parameter set information, information pertaining to ROI or any other information needed to characterize the sub-samples. For parameter sets, the sub-sample description association table 724 indicates the parameter set associated with each sub-sample. In such a case, the sub-sample description ID corresponds to the parameter set identifier. Similarly, a sub-sample can represent different regions-of-interest as follows. Define a sub-sample as one or more coded macroblocks and then use the sub-sample description association table to represent the division of the coded microblocks of a video frame or image into different regions. For example, the coded macroblocks in a frame can be divided into foreground and background macroblocks with two sub-sample description ID (e.g., sub-sample description IDs of 1 and 2), indicating assignment to the foreground and background regions, respectively.

Figure 7F illustrates different types of sub-samples. A sub-sample may represent a slice 732 with no partition, a slice 734 with multiple data partitions, a header 736 within a slice, a data partition 738 in the middle of a slice, the last data partition 740 of a slice, an SEI information packet 742, etc. Each of these sub-sample types may be associated with a specific value of an 8-bit mask 744 shown in **Figure 7G**. The 8-bit mask may form the 8 least significant bits of the 32-bit sub-sample description ID field as discussed above. **Figure 7H** illustrates the sub-sample description association box 724 having the description type identifier equal to "jvtd". The table 726 includes the 32-bit sub-sample description ID field storing the values illustrated in **Figure 7G**.

Figures 7H – 7K illustrate compression of data in a sub-sample description association table.

Referring to **Figure 7I**, an uncompressed table 726 includes a sequence 750 of sub-sample description IDs that repeats a sequence 748. In a compressed table 746, the repeated sequence 750 has been compressed into a reference to the sequence 748 and the number of times this sequence occurs.

In one embodiment illustrated in **Figure 7J**, a sequence occurrence can be encoded in the sub-sample description ID field by using its most significant bit as a run of sequence flag 754, its next 23 bits as an occurrence index 756, and its less significant bits as an occurrence length 758. If the flag 754 is set to 1, then it indicates that this entry is an occurrence of a repeated sequence. Otherwise, this entry is a sub-sample description ID. The occurrence index 756 is the index in the sub-sample description association box 724 of the first occurrence of the sequence, and the length 758 indicates the length of the repeated sequence occurrence.

In another embodiment illustrated in **Figure 7K**, a repeated sequence occurrence table 760 is used to represent the repeated sequence occurrence. The most significant bit of the sub-sample description ID field is used as a run of sequence flag 762 indicating whether the entry is a sub-sample description ID or a sequence index 764 of the entry in the repeated sequence occurrence table 760 that is part of the sub-sample description association box 724. The repeated sequence occurrence table 760 includes an occurrence

index field to specify the index in the sub-sample description association box 724 of the first item in the repeated sequence and a length field to specify the length of the repeated sequence.

Parameter Sets

In certain media formats, such as JVT, the "header" information containing the critical control values needed for proper decoding of media data are separated/decoupled from the rest of the coded data and stored in parameter sets. Then, rather than mixing these control values in the stream along with coded data, the coded data can refer to necessary parameter sets using a mechanism such as a unique identifier. This approach decouples the transmission of higher level coding parameters from coded data. At the same time, it also reduces redundancies by sharing common sets of control values as parameter sets.

To support efficient transmission of stored media streams that use parameter sets, a sender or player must be able to quickly link coded data to a corresponding parameter in order to know when and where the parameter set must be transmitted or accessed. One embodiment of the present invention provides this capability by storing data specifying the associations between parameter sets and corresponding portions of media data as parameter set metadata in a media file format.

Figures 8 and 9 illustrate processes for storing and retrieving parameter set metadata that are performed by the encoding system 100 and the decoding system 200 respectively. The processes may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both.

Figure 8 is a flow diagram of one embodiment of a method 800 for creating parameter set metadata at the encoding system 100. Initially, method 800 begins with processing logic receiving a file with encoded media data (processing block 802). The file includes sets of encoding parameters that specify how to decode portions of the media data. Next, processing logic examines the relationships between the sets of encoding

parameters referred to as parameter sets and the corresponding portions of the media data (processing block 804) and creates parameter set metadata defining the parameter sets and their associations with the media data portions (processing block 806). The media data portions may be represented by samples or sub-samples.

In one embodiment, the parameter set metadata is organized into a set of predefined data structures (e.g., a set of boxes). The set of predefined data structures may include a data structure containing descriptive information about the parameter sets and a data structure containing information that defines associations between samples and corresponding parameter sets. In one embodiment, the set of predefined data structures also includes a data structure containing information that defines associations between sub-samples and corresponding parameter sets. The data structures containing sub-sample to parameter set association information may or may not override the data structures containing sample to parameter set association information.

Next, in one embodiment, processing logic determines whether any parameter set data structure contains a repeated sequence of data (decision box 808). If this determination is positive, processing logic converts each repeated sequence of data into a reference to a sequence occurrence and the number of times the sequence occurs (processing block 810).

Afterwards, at processing block 812, processing logic includes the parameter set metadata into a file associated with media data using a specific media file format (e.g., the JVT file format). Depending on the media file format, the parameter set metadata may be stored with track metadata and/or sample metadata (e.g., the data structure containing descriptive information about parameter sets may be included in a track box and the data structure(s) containing association information may be included in a sample table box) or independently from the track metadata and/or sample metadata.

Figure 9 is a flow diagram of one embodiment of a method 900 for utilizing parameter set metadata at the decoding system 200. Initially, method 900 begins with processing logic receiving a file associated with encoded media data (processing block 902). The file may be received from a database (local or external), the encoding system 100, or

from any other device on a network. The file includes parameter set metadata that defines parameter sets for the media data and associations between the parameter sets and corresponding portions of the media data (e.g., corresponding samples or sub-samples).

Next, processing logic extracts the parameter set metadata from the file (processing block 904). As discussed above, the parameter set metadata may be stored in a set of data structures (e.g., a set of boxes).

Further, at processing block 906, processing logic uses the extracted metadata to determine which parameter set is associated with a specific media data portion (e.g., a sample or a sub-sample). This information may then be used to control transmission time of media data portions and corresponding parameter sets. That is, a parameter set that is to be used to decode a specific sample or sub-sample must be sent prior to a packet containing the sample or sub-sample or with the packet containing the sample or sub-sample.

Accordingly, the use of parameter set metadata enables independent transmission of parameter sets on a more reliable channel, reducing the chance of errors or data loss causing parts of the media stream to be lost.

Exemplary parameter set metadata structures will now be described with reference to an extended ISO media file format (referred to as an extended ISO). It should be noted, however, that other media file formats can be extended to incorporate various data structures for storing parameter set metadata.

Figures 10A – 10E illustrate exemplary data structures for storing parameter set metadata.

Referring to **Figure 10A**, a track box 1002 that contains track metadata boxes defined by the ISO file format is extended to include a parameter set description box 1004. In addition, a sample table box 1006 that contains sample metadata boxes defined by ISO file format is extended to include a sample to parameter set box 1008. In one embodiment, the sample table box 1006 includes a sub-sample to parameter set box which may override the sample to parameter set box 1008 as will be discussed in more detail below.

In one embodiment, the parameter set metadata boxes 1004 and 1008 are mandatory. In another embodiment, only the parameter set description box 1004 is mandatory. In yet another embodiment, all of the parameter set metadata boxes are optional.

Referring to **Figure 10B**, a parameter set description box 1010 contains a version field that specifies the version of the parameter set description box 1010, a parameter set description count field to provide the number of entries in a table 1012, and a parameter set entry field containing entries for the parameter sets themselves.

Parameter sets may be referenced from the sample level or the sub-sample level. Referring to **Figure 10C**, a sample to parameter set box 1014 provides references to parameter sets from the sample level. The sample to parameter set box 1014 includes a version field that specifies the version of the sample to parameter set box 1014, a default parameter set ID field that specifies the default parameter set ID, an entry count field that provides the number of entries in the table 1016. Each entry in table 1016 contains a first sample field providing the index of a first sample in a run of samples that share the same parameter set, and a parameter set index specifying the index to the parameter set description box 1010. If the default parameter set ID is equal to 0, then the samples have different parameter sets that are stored in the table 1016. Otherwise, a constant parameter set is used and no array follows.

In one embodiment, data in the table 1016 is compressed by converting each repeated sequence into a reference to an initial sequence and the number of times this sequence occurs, as discussed in more detail above in conjunction with the sub-sample description association table.

Parameter sets may be referenced from the sub-sample level by defining associations between parameter sets and sub-samples. In one embodiment, the associations between parameter sets and sub-samples are defined using a sub-sample description association box described above. **Figure 10D** illustrates a sub-sample description association box 1018 with the description type identifier referring to parameter sets (e.g., the description type identifier is equal to "pars"). Based on this description

type identifier, the sub-sample description ID in the table 1020 indicates the index in the parameter set description box 1010.

In one embodiment, when the sub-sample description association box 1018 with the description type identifier referring to parameter sets is present, it overrides the sample to parameter set box 1014.

A parameter set may change between the time the parameter set is created and the time the parameter set is used to decode a corresponding portion of media data. If such a change occurs, the decoding system 200 receives a parameter update packet specifying a change to the parameter set. The parameter set metadata includes data identifying the state of the parameter set both before the update and after the update.

Referring to **Figure 10E**, the parameter set description box 1010 includes an entry for the initial parameter set 1022 created at time t_0 and an entry for an updated parameter set 1024 created in response to a parameter update packet 1026 received at time t_1. The sub-sample description association box 1018 associates the two parameter sets with corresponding sub-samples.

Sample Groups

While the samples within a track can have various logical groupings (partitions) of samples into sequences (possibly non-consecutive) that represent high-level structures in the media data, existing file formats do not provide convenient mechanisms for representing and storing such groupings. For example, advanced coding formats such as JVT organize samples within a single track into groups based on their inter-dependencies. These groups (referred to herein as sequences or sample groups) may be used to identify chains of disposable samples when required by network conditions, thus supporting temporal scalability. Storing metadata that defines sample groups in a file format enables the sender of the media to easily and efficiently implement the above features.

An example of a sample group is a set of samples whose inter-frame dependencies allow them to be decoded independently of other samples. In JVT, such a sample group is referred to as an enhanced group of pictures (enhanced GOP). In an enhanced GOP,

samples may be divided into sub-sequences. Each sub-sequence includes a set of samples that depend on each other and can be disposed of as a unit. In addition, samples of an enhanced GOP may be hierarchically structured into layers such that samples in a higher layer are predicted only from samples in a lower layer, thus allowing the samples of the highest layer to be disposed of without affecting the ability to decode other samples. The lowest layer that includes samples that do not depend on samples in any other layers is referred to as a base layer. Any other layer that is not the base layer is referred to as an enhancement layer.

Figure 11 illustrates an exemplary enhanced GOP in which the samples are divided into two layers, a base layer 1102 and an enhancement layer 1104, and two sub-sequences 1106 and 1108. Each of the two sub-sequences 1106 and 1108 can be dropped independently of each other.

Figures 12 and 13 illustrate processes for storing and retrieving sample group metadata that are performed by the encoding system 100 and the decoding system 200 respectively. The processes may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both.

Figure 12 is a flow diagram of one embodiment of a method 1200 for creating sample group metadata at the encoding system 100. Initially, method 1200 begins with processing logic receiving a file with encoded media data (processing block 1202). Samples within a track of the media data have certain inter-dependencies. For example, the track may include I-frames that do not depend on any other samples, P-frames that depend on a single prior sample, and B-frames that depend on two prior samples including any combination of I-frames, P-frames and B-frames. Based on their inter-dependencies, samples in a track can be logically combined into sample groups (e.g., enhanced GOPs, layers, sub-sequences, etc.).

Next, processing logic examines the media data to identify sample groups in each track (processing block 1204) and creates sample group metadata that describes the sample groups and defines which samples are contained in each sample group (processing block

1206). In one embodiment, the sample group metadata is organized into a set of predefined data structures (e.g., a set of boxes). The set of predefined data structures may include a data structure containing descriptive information about each sample group, a data structure containing information that identifies samples contained in each sample group, a data structure containing information that describes sub-sequences, and a data structure containing information that describes layers. Next, in one embodiment, processing logic determines whether any sample group data structure contains a repeated sequence of data (decision box 1208). If this determination is positive, processing logic converts each repeated sequence of data into a reference to a sequence occurrence and the number of times the sequence occurs (processing block 1210).

Afterwards, at processing block 1212, processing logic includes the sample group metadata into a file associated with media data using a specific media file format (e.g., the JVT file format). Depending on the media file format, the sample group metadata may be stored with sample metadata (e.g., the sample group data structures may be included in a sample table box) or independently from the sample metadata.

Figure 13 is a flow diagram of one embodiment of a method 1300 for utilizing sample group metadata at the decoding system 200. Initially, method 1300 begins with processing logic receiving a file associated with encoded media data (processing block 1302). The file may be received from a database (local or external), the encoding system 100, or from any other device on a network. The file includes sample group metadata that defines sample groups in the media data.

Next, processing logic extracts the sample group metadata from the file (processing block 1304). As discussed above, the sample group metadata may be stored in a set of data structures (e.g., a set of boxes).

Further, at processing block 1306, processing logic uses the extracted sample group metadata to identify chains of samples that can be disposed of without affecting the ability to decode other samples. In one embodiment, this information may be used to access samples in a specific sample group and determine which samples can be dropped in response to a change in network capacity. In other embodiments, sample group metadata

is used to filter samples so that only a portion of the samples in a track are processed or rendered.

Accordingly, the sample group metadata facilitates selective access to samples and scalability.

Exemplary sample group metadata structures will now be described with reference to an extended ISO media file format (referred to as an extended MP4). It should be noted, however, that other media file formats can be extended to incorporate various data structures for storing sample group metadata.

Figures 14A – 14E illustrate exemplary data structures for storing sample group metadata.

Referring to **Figure 14A**, a sample table box 1400 that contains sample metadata boxes defined by MP4 is extended to include a sample group box 1402 and a sample group description box 1404. In one embodiment, the sample group metadata boxes 1402 and 1404 are optional. In one embodiment (not shown), the sample table box 1400 includes additional optional sample group metadata boxes such as a sub-sequence description entry box and a layer description entry box.

Referring to **Figure 14B**, a sample group box 1406 is used to find a set of samples contained in a particular sample group. Multiple instances of the sample group box 1406 are allowed to correspond to different types of sample groups (e.g., enhanced GOPs, sub-sequences, layers, parameter sets, etc.). The sample group box 1406 contains a version field that specifies the version of the sample group box 1406, an entry count field to provide the number of entries in a table 1408, a sample group identifier field to identify the type of the sample group, a first sample field providing the index of a first sample in a run of samples that are contained in the same sample group, and a sample group description index specifying the index to a sample group description box.

Referring to **Figure 14C**, a sample group description box 1410 provides information about the characteristics of a sample group. The sample group description box 1410 contains a version field that specifies the version of the sample group description box 1410, an entry count field to provide the number of entries in a table 1412, a sample group

identifier field to identify the type of the sample group, and a sample group description field to provide sample group descriptors.

Referring to **Figure 14D**, the use of the sample group box 1416 for the layers ("layr") sample group type is illustrated. Samples 1 through 11 are divided into three layers based on the samples' inter-dependencies. In layer 0 (the base layer), samples (samples 1, 6 and 11) depend only on each other but not on samples in any other layers. In layer 1, samples (samples 2, 5, 7, 10) depend on samples in the lower layer (i.e., layer 0) and samples within this layer 1. In layer 2, samples (samples 3, 4, 8, 9) depend on samples in lower layers (layers 0 and 1) and samples within this layer 2. Accordingly, the samples of layer 2 can be disposed of without affecting the ability to decode samples from lower layers 0 and 1.

Data in the sample group box 1416 illustrates the above associations between the samples and the layers. As shown, this data includes a repetitive layer pattern 1414 which can be compressed by converting each repeated layer pattern into a reference to an initial layer pattern and the number of times this pattern occurs, as discussed in more detail above.

Referring to **Figure 14E**, the use of a sample group box 1418 for the sub-sequence ("sseq") sample group type is illustrated. Samples 1 through 11 are divided into four sub-sequences based on the samples' inter-dependencies. Each sub-sequence, except sub-sequence 0 at layer 0, includes samples on which no other sub-sequences depend. Thus, the samples in the sub-sequence can be disposed of as a unit when needed.

Data in the sample group box 1418 illustrates associations between the samples and the sub-sequences. This data allows random access to samples at the beginning of a corresponding sub-sequence.

In one embodiment, a sub-sequence description entry box is used to describe each sub-sequence of samples in a GOP. The sub-sequence description entry box provides dependency information pertaining to sub-sequence identifier data, average bit rate data, average frame rate data, a reference number data, and an array containing information about the referenced data.

The dependency information identifies a sub-sequence that is used as a reference for the sub-sequence described in this entry. The sub-sequence identifier data provides an identifier of the sub-sequence described in this entry. The average bit rate data contains the average bit rate (e.g., in bits or seconds) of this sub-sequence. In one embodiment, the calculation of the average bit rate takes into account payloads and payload headers. In one embodiment, the average bit rate is equal to zero if the average bit rate is undefined.

The average frame rate data contains the average frame rate in frames of the entry's sub-sequence. In one embodiment, the average frame rate is equal to zero if the average frame rate is undefined.

The reference number data provides the number of directly referenced sub-sequences in the entry's sub-sequence. The array of referenced data provides the identification information of the referenced sub-sequences.

In one embodiment, an additional layer description entry box is used to provide layer information. The layer description entry box provides the number of the layer, the average bit rate of the layer, and the average frame rate. The number of the layer may be equal to zero for the base layer and one or higher for each enhancement layer. The average bit rate may be equal to zero when the average bit rate is undefined, and the average frame rate may be equal to zero when the average frame rate is undefined.

Stream Switching

In typical streaming scenarios, one of the key requirements is to scale the bit rate of the compressed data in response to changing network conditions. The simplest way to achieve this is to encode multiple streams with different bit-rates and quality settings for representative network conditions. The server can then switch amongst these pre-coded streams in response to network conditions.

The JVT standard provides a new type of picture, called switching pictures that allow one picture to reconstruct identically to another without requiring the two pictures to use the same frame for prediction. In particular, JVT provides two types of switching pictures: SI-pictures, which, like I-frames, are coded independent of any other pictures;

and SP-pictures, which are coded with reference to other pictures. Switching pictures can be used to implement switching amongst streams with different bit-rates and quality setting in response to changing delivery conditions, to provide error resilience, and to implement trick modes like fast forward and rewind.

However, to use JVT switching pictures effectively when implementing stream switching, error resilience, trick modes, and other features, the player has to know which samples in the stored media data have the alternate representations and what their dependencies are. Existing file formats do not provide such capability.

One embodiment of the present invention addresses the above limitation by defining switch sample sets. A switch sample set represents a set of samples whose decoded values are identical but which may use different reference samples. A reference sample is a sample used to predict the value of another sample. Each member of a switch sample set is referred to as a switch sample. **Figure 15A** illustrate the use of a switch sample set for bit stream switching.

Referring to **Figure 15A**, stream 1 and stream 2 are two encodings of the same content with different quality and bit-rate parameters. Sample S12 is a SP-picture, not occurring in either stream, that is used to implement switching from stream 1 to stream 2 (switching is a directional property). Samples S12 and S2 are contained in a switch sample set. Both S1 and S12 are predicted from sample P12 in track 1 and S2 is predicted from sample P22 in track 2. Although samples S12 and S2 use different reference samples, their decoded values are identical. Accordingly, switching from stream 1 to stream 2 (at sample S1 in stream 1 and S2 in stream 2) can be achieved via switch sample S12.

Figures 16 and 17 illustrate processes for storing and retrieving switch sample metadata that are performed by the encoding system 100 and the decoding system 200 respectively. The processes may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both.

Figure 16 is a flow diagram of one embodiment of a method 1600 for creating switch sample metadata at the encoding system 100. Initially, method 1600 begins with

processing logic receiving a file with encoded media data (processing block 1602). The file includes one or more alternate encodings for the media data (e.g., for different bandwidth and quality settings for representative network conditions). The alternate encodings includes one or more switching pictures. Such pictures may be included inside the alternate media data streams or as separate entities that implement special features such as error resilience or trick modes. The method for creating these tracks and switch pictures is not specified by this invention but various possibilities would be obvious to one versed in the art. For example, the periodic (e.g., every 1 second) placement of switch samples between each pair of tracks containing alternate encodings.

Next, processing logic examines the file to create switch sample sets that include those samples having the same decoding values while using different reference samples (processing block 1604) and creates switch sample metadata that defines switch sample sets for the media data and describes samples within the switch sample sets (processing block 1606). In one embodiment, the switch sample metadata is organized into a predefined data structure such as a table box containing a set of nested tables.

Next, in one embodiment, processing logic determines whether the switch sample metadata structure contains a repeated sequence of data (decision box 1608). If this determination is positive, processing logic converts each repeated sequence of data into a reference to a sequence occurrence and the number of times the sequence occurs (processing block 1610).

Afterwards, at processing block 1612, processing logic includes the switch sample metadata into a file associated with media data using a specific media file format (e.g., the JVT file format). In one embodiment, the switch sample metadata may be stored in a separate track designated for stream switching. In another embodiment, the switch sample metadata is stored with sample metadata (e.g., the sequences data structures may be included in a sample table box).

Figure 17 is a flow diagram of one embodiment of a method 1700 for utilizing switch sample metadata at the decoding system 200. Initially, method 1700 begins with processing logic receiving a file associated with encoded media data (processing block

1702). The file may be received from a database (local or external), the encoding system 100, or from any other device on a network. The file includes switch sample metadata that defines switch sample sets associated with the media data.

Next, processing logic extracts the switch sample metadata from the file (processing block 1704). As discussed above, the switch sample metadata may be stored in a data structure such as a table box containing a set of nested tables.

Further, at processing block 1706, processing logic uses the extracted metadata to find a switch sample set that contains a specific sample and select an alternative sample from the switch sample set. The alternative sample, which has the same decoding value as the initial sample, may then be used to switch between two differently encoded bit streams in response to changing network conditions, to provide random access entry point into a bit stream, to facilitate error recovery, etc.

An exemplary switch sample metadata structure will now be described with reference to an extended ISO media file format (referred to as an extended MP4). It should be noted, however, that other media file formats could be extended to incorporate various data structures for storing switch sample metadata.

Figure 18 illustrates an exemplary data structure for storing switch sample metadata. The exemplary data structure is in the form of a switch sample table box that includes a set of nested tables. Each entry in a table 1802 identifies one switch sample set. Each switch sample set consists of a group of switch samples whose reconstruction is objectively identical (or perceptually identical) but which may be predicted from different reference samples that may or may not be in the same track (stream) as the switch sample. Each entry in the table 1802 is linked to a corresponding table 1804. The table 1804 identifies each switch sample contained in a switch sample set. Each entry in the table 1804 is further linked to a corresponding table 1806 which defines the location of a switch sample (i.e., its track and sample number), the track containing reference samples used by the switch sample, the total number of reference samples used by the switch sample, and each reference sample used by the switch sample.

As illustrated in **Figure 15A**, in one embodiment, the switch sample metadata may be used to switch between differently encoded versions of the same content. In MP4, each alternate coding is stored as a separate MP4 track and the "alternate group" in the track header indicates that it is an alternate encoding of specific content.

Figure 15B illustrates a table containing metadata that defines a switch sample set 1502 consisting of samples S2 and S12 according to **Figure 15A**.

Figure 15C is a flow diagram of one embodiment of a method 1510 for determining a point at which a switch between two bit streams is to be performed. Assuming that the switch is to be performed from stream 1 to stream 2, method 1510 begins with searching switch sample metadata to find all switch sample sets that contain a switch sample with a reference track of stream 1 and a switch sample with a switch sample track of stream 2 (processing block 1512). Next, the resulting switch sample sets are evaluated to select a switch sample set in which all reference samples of a switch sample with the reference track of stream 1 are available (processing block 1514). For example, if the switch sample with the reference track of stream 1 is a P frame, one sample before switching is required to be available. Further, the samples in the selected switch sample set are used to determine the switching point (processing block 1516). That is, the switching point is considered to be immediately after the highest reference sample of the switch sample with the reference track of stream 1, via the switch sample with the reference track of stream 1, and to the sample immediately following the switch sample with the switch sample track of stream 2.

In another embodiment, switch sample metadata may be used to facilitate random access entry points into a bit stream as illustrated in **Figures 19A – 19C**.

Referring to **Figures 19A and 19B**, a switch sample 1902 consists of samples S2 and S12. S2 is a P-frame predicted from P22 and used during usual stream playback. S12 is used as a random access point (e.g., for splicing). Once S12 is decoded, stream playback continues with decoding of P24 as if P24 was decoded after S2.

Figure 19C is a flow diagram of one embodiment of a method 1910 for determining a random access point for a sample (e.g., sample S on track T). Method 1910 begins with searching switch sample metadata to find all switch sample sets that contain a switch

sample with a switch sample track T (processing block 1912). Next, the resulting switch sample sets are evaluated to select a switch sample set in which a switch sample with the switch sample track T is the closest sample prior to sample S in decoding order (processing block 1914). Further, a switch sample (sample SS) other than the switch sample with the switch sample track T is chosen from the selected switch sample set for a random access point to sample S (processing block 1916). During stream playback, sample SS is decoded (following by the decoding of any reference samples specified in the entry for sample SS) instead of sample S.

In yet another embodiment, switch sample metadata may be used to facilitate error recovery as illustrated in **Figures 20A – 20C**.

Referring to **Figures 20A and 20B**, a switch sample 2002 consists of samples S2, S12 and S22. Sample S2 is predicted from sample P4. Sample S12 is predicted from sample S1. If an error occurs between samples P2 and P4, the switch sample S12 can be decoded instead of sample S2. Streaming then continues with sample P6 as usual. If an error affects sample S1 as well, switch sample S22 can be decoded instead of sample S2, and then streaming will continue with sample P6 as usual.

Figure 20C is a flow diagram of one embodiment of a method 2010 for facilitating error recovery when sending a sample (e.g., sample S). Method 2010 begins with searching switch sample metadata to find all switch sample sets that contain a switch sample equal to sample S or following sample S in the decoding order (processing block 2012). Next, the resulting switch sample sets are evaluated to select a switch sample set with a switch sample SS that is the closest to sample S and whose reference samples are known (via feedback or some other information source) to be correct (processing block 2014). Further, switch sample SS is sent instead of sample S (processing block 2016).

Storage of Parameter Sets and Supplemental Enhancement Information.

As discussed above, some metadata such as parameter set metadata may be stored separately from the associated media data. **Figure 21** illustrates separate storage of parameter set metadata, according to one embodiment of the present invention.

Referring to **Figure 21**, the media data is stored in a video track 2102 and the parameter set metadata is stored in a separate parameter track 2104 which may be marked as "inactive" to indicate that it does not store media data. Timing information 2106 provides synchronization between the video track 2102 and the parameter track 2104. In one embodiment, the timing information is stored in a sample table box of each of the video track 2102 and the parameter set track 2104. In one embodiment, each parameter set is represented by one parameter set sample, and the synchronization is achieved if the timing information of a media sample is equal to the timing information of a parameter set sample.

In another embodiment, object descriptor (OD) messages are used to include parameter set metadata. According to the MPEG-4 standards, an object descriptor represents one or more elementary stream descriptors that provide configuration and other information for the streams that relate to a single object (media object or scene description). Object descriptor messages are sent in an object descriptor stream. As illustrated in **Figure 22**, parameter sets are included as object descriptor messages 2204 into an object descriptor stream 2202. The object descriptor stream 2202 is synchronized with a video elementary stream carrying the media data.

Storage of SEI will now be discussed in more detail.

In one embodiment, SEI data is stored in the elementary stream with the media data. **Figure 23** illustrates a SEI message 2304 embedded directly in elementary stream data 2303 along with the media data.

In another embodiment, SEI messages are stored as samples in a separate SEI track. **Figures 24 and 25** illustrate storage of SEI messages in a separate track, according to some embodiments of the present invention.

Referring to **Figure 24**, media data is stored in a video track 2402 and SEI messages are stored in a separate SEI track 2404 as samples. Timing information 2406 provides synchronization between the video track 2402 and the SEI track 2404.

Referring to **Figure 25**, media data is stored in a video track 2502 and SEI messages are stored in an object content information (OCI) track 2504. Timing information 2506

provides synchronization between the video track 2502 and the OCI track 2504. According to the MPEG-4 standards, the OCI track 2504 is designated to store OCI data that is commonly used to provide textual descriptive information about scene events. Each SEI message is stored in the OCI track 2504 as an object descriptor. In one embodiment, an OCI descriptor element field that typically specifies the type of data stored in the OCI track is used to carry SEI messages.

In yet another embodiment, SEI data is stored as metadata separate from the media data. **Figure 26** illustrates storage of SEI data as metadata, according to one embodiment of the present invention.

Referring to **Figure 26**, a user data box 2602 defined by the ISO Media File Format is used to store SEI messages. Specifically, each SEI message is stored in a SEI user data box 2604 in the user data box 2602 that is contained in a track or a movie box.

In one embodiment, the metadata included in the SEI messages contains descriptions of the media data. These descriptions may represent descriptors and description schemes that are defined by the MPEG-7 standards. In one embodiment, SEI messages support the inclusion of XML-based data such as XML-based descriptions. In addition, the SEI messages support registration of different types of enhancement information. For example, the SEI messages may support anonymous user data without registering a new type. Such data may be intended to be private to a particular application or organization. In one embodiment, the presence of SEI is indicated in a bitstream environment by a designated start code.

In one embodiment, the capability of a decoder to provide any or all of the enhanced capabilities described in a SEI message is signaled by external means (e.g., Recommendation H.245 or SDP). Decoders that do not provide the enhanced capabilities may simply discard SEI messages.

In one embodiment, the synchronization of media data (e.g., video coding layer data) and SEI messages containing descriptions of the media data is provided using designated fields in a payload header of SEI messages, as will be discussed in more detail below.

In one embodiment, Network Adaptation Layers support a means to carry supplemental enhancement information messages in the underlying transport systems. Network adaptation may allow either an in-band (in the same transport stream as the video coding layer) or out-of-band means for signaling SEI messages.

In one embodiment, the inclusion of MPEG-7 metadata into SEI messages is achieved by using SEI as a delivery layer for MPEG-7 metadata. In particular, an SEI message encapsulates an MPEG-7 Systems Access Unit (Fragment) that represents **one or more description fragments**. The synchronization of MPEG- 7 Access Units with the media data may be provided using designated fields in a payload header of SEI messages.

In another embodiment, the inclusion of MPEG-7 metadata into SEI messages is achieved by allowing description units to be sent in SEI messages in either a text or a binary encoding. A description unit may be a single MPEG-7 descriptor or description scheme and may be used to represent partial information from a complete description. For example, the following shows the XML syntax for a scalable color descriptor:

```
<Mpeg7>
    <DescriptionUnit xsi:type="ScalableColorType" numOfCoeff="16"
        numOfBitplanesDiscarded="O" >
    <Coeff> 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 </Coeff>
    </DescriptionUnit>
</Mpeg7>
```

The descriptors or description scheme instances may be associated with corresponding portions of the media data (e.g., sub-samples, samples, fragments, etc.) through the SEI message header, as will be discussed in greater detail below. This embodiment allows, for example, a binary or textually encoded color descriptor for a single frame to be sent as an SEI message. Using SEI messages, an implicit description of the video coding stream can be provided. An implicit description is a complete description of the video coding stream in which the description units are implicitly contained. An implicit description may have the following form:

```
<Mpeg7>
```

```
<Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="VideoType">
        <Video>
            <CreationInformation>
                <Creation>
                    <Title> Worldcup Soccer </Title>
                </Creation>
            </CreationInformation>
            <MediaTime>
                <MediaTimePoint>TOO:OO:OO</MediaTi
                mePoint>
                <MediaDuration>PT1M30S</MediaDuratio
                n>
            </MediaTime>
            <VisualDescriptor xsi:type="GoFGoPColorType"
            aggregation="Average">
                <ScalableColor numOfCoeff="16"
            numOfBitplanesDiscarded="O">
                    <Coeff> 123 4 567 8 9 0 1 2 3 4 5 6
                </Coeff>
                </ScalableColor>
            </VisualDescriptor>
        </Video>
    </MultimediaContent >
</Description>
</Mpeg7>
```

In one embodiment, a revised format for SEI is provided to support the inclusion of descriptions into SEI messages. Specifically, SEI is represented as a group of SEI messages. In one embodiment, SEI is encapsulated into chunks of data. Each SEI chunk may contain one or more SEI messages. Each SEI message contains a SEI header and a SEI payload. The SEI header starts at a byte-aligned position from the first byte of a SEI chunk or from the first byte after the previous SEI message. The payload immediately follows the SEI header starting on the byte following the SEI header.

The SEI header includes message type, optional identifiers of media data portions (e.g., a sub-sample, a sample, and a fragment), and the payload length. The syntax of the SEI header may be as follows:

```
aligned(8) SupplementalEnhancementInformation
{
        aligned unsigned int(13) MessageType;
        aligned unsigned int(2} MessageScope
                if (MessageScope = = O)
                {
                        // Message is related to a sample
                        unsigned int(16) SampleID;
                }
                else
                {
                        // Reserved
                }
        aligned unsigned int(16) PayloadLength;
        aligned unsigned int(8) Payload[PayloadLength];
}
```

The MessageType field indicates the type of message in the payload. Exemplary SEI message type codes are specified in Table 1 as follows:

Message Code	Picture Message	Slice Message	Message Description
MPEG-7			
			MPEG-7 Binary Access Unit
			MPEG-7 Textual Access Unit
			MPEG-7 JVT Metadata D/DS Fragment Text
			MPEG-7 JVT Metadata D/DS Fragment Binary
New Types			Arbitrary XMLxxxMessage
			JVT Specified XML message.

H.263 Annex I			
			Video Time Segment Start Tag
			Video Time Segment End Tag
H.263L Annex W			
0			Arbitrary Binary Data
1			Arbitrary Text
2			Copyright Text
3			Caption Text
4			Video Description Text Human readable text.
5			Uniform Resource Identifier Text
6			Current Picture Header Repetition
7			Previous Picture Header Repetition
8			Next Picture Header Repetition, Reliable TR
9			Next Picture Header Repetition, Unreliable TR
10			Top Interlaced Field Indication
11			Bottom Interlaced Field Indication
12			Picture Number
13			Spare Reference Pictures

Table 1

The PayloadLength field specifies the length of the SEI message payload in bytes. The SEI header also includes a sample synchronization flag indicating whether this SEI message is associated with a particular sample and a sub-sample synchronization flag indicating whether this SEI message is associated with a particular sub-sample (if sub-sample synchronization flag is set, the sample synchronization flag is also set). The SEI

payload further includes an optional sample identifier field specifying the sample that this message is associated with and an optional sub-sample identifier field specifying the sub-sample that the message is associated with. The sample identifier field is present only if the sample synchronization flag is set. Similarly, the sub-sample identifier field is present only if the sub-sample synchronization flag is set. The sample identifier and sub-sample identifier fields allow synchronization of the SEI message with the media data.

In one embodiment, each SEI message is sent in a SEI message descriptor. SEI descriptors are encapsulated into SEI units that contain one or more SEI messages. The syntax of a SEI message unit is as follows:

```
aligned(8) class SEIMessageUnit
{
        SEIMessageDescriptor descriptor[0. .255];
}
```

The syntax of a SEI message descriptor is as follows:

```
abstract expandable(2**16-1) aligned(8) class SEIMessageDescriptor

        : tag unsigned int(16)

{
        unsigned int(16)  type = tag;
}
```

The type field indicates the type of an SEI message. Exemplary types of SEI messages are provided in Table 2 as follows:

Tag Value	Tag name	
0x0000	Forbidden	
0x0000	Associate Information SEI	
	SEIMetadataDescriptorTag	
	SEIMetadataRefDescriptorTa	

	g	
	SEITextDescriptorTag	
	SEIXMLDescriptorTag	
	SEIStartSegmentTag	
	SEIEndSegmentTag	
-0x6FFF	Reserved for ISO use	
0x7000-FFF	Reserved for application use.	
0x8000-FFFF	Reserved for assignment by a SC29 Registration Authority.	

Table 2

SEI messages of various types illustrated in Table 2 will now be described in more detail.

The SEIXMLDescriptor type refers to a descriptor that encapsulates XML-based data which may include, for example, a complete XML document or an XML fragment from a larger document. The syntax of SEIXMLDescriptor is as follows:

```
class SEIXMLDescriptor : SEIMessageDescriptor(SEIXMLDescriptorTag)

{
        unsigned int(8) xmlData[] ;
{
```

The SEIMetadataDescriptor type refers to a descriptor that contains metadata. The syntax of SEIMetadataDescriptor is as follows:

```
class SEIMetadataDescriptor : SEIMessageDescriptor (SEIXMLDescriptorTag)

{
        unsigned int(8)         metadataFormat;
        unsigned int(8)         metadataContent[];
}
```

The metadataFormat field identifies the format of the metadata. Exemplary values of the metadata format are illustrated in Table 3 as follows:

Value	Description
0x00-0x0F	Reserved
0x10	ISO 15938 (MPEG-7) defined
0x11-0x3F	Reserved
0x40-0xFF	Registration Authority defined

Table 3

The values 0x10 identifies MPEG-7 defined data. The values in the inclusive range of 0x40 up to 0xFF are available to signal the use of private formats.

The metadataContent field contains the representation of the metadata in the format specified by the metadataFormat field.

The SEIMetadataRefDescriptor type refers to a descriptor that specifies a URL pointing to the location of metadata. The syntax of SEIMetadatRefDescriptor is as follows:

```
class SEIMetadataRefDescriptor:
SEIMessageDescriptor(SEIMetdataRefDescriptorTag)
{
        bit (8) URLString [] ;
}
```

The URLString field contains a UTF-8 encoded URL that points to the location of metadata.

The SEITextDescriptor type refers to a descriptor that contains text describing, or pertaining to, the video content. The syntax of SEITextDescriptor is as follows:

```
class SEIMessageDescriptor : SEIMessageDescriptor(SEIXMLDescriptorTag)
{
        unsigned int(24)        languageCode;
        unsigned int(8)         text[];
}
```

The languageCode field contains the language code of the language of the following text fields. The text field contains the UTF-8 encoded textual data.

The SEIURIDescriptor type refers to a descriptor that contains a uniform resource identifier (URI) related to the video content. The syntax of SEIURIDescriptor is as follows:

```
class SEIURIDescriptor : SEIMessageDescriptor(SEIURIDescriptorTag)
{
        unsigned int (16)    uriString[];
}
```

The uriString field contains a URI of the video content.

The SEIOCIDescriptor type refers to a descriptor that contains an SEI message that represents an Object Content Information (OCI) descriptor. The syntax of SEIOCIDescriptor is as follows:

```
class SEIOCIDescriptor : SEIMessageDescriptor(SEIOCIDescriptorTag)
{
        OCI_Descriptor        ociDescr;
}
```

The ociDescr field contains an OCI descriptor.

The SEIStartSegmentDescriptor type refers to a descriptor that indicates the start of a segment, which may then be referenced in other SEI messages. The segment start is associated with a certain layer (e.g., a group of samples, segment, sample, or sub-sample) to which this SEI descriptor is applied. The syntax of SEIStartSegmentDecriptor is as follows:

```
class SEIStartSegmentDescriptor :
        SEIMessageDescriptor(SEIStartSegmentDescriptorTag)
{
        unsigned int(32)        segmentID;
}
```

The segmentID field indicates a unique binary identifier within this stream for the segment. This value may be used to reference the segment in other SEI messages.

The SEIEndSegmentDecriptor type refers to a descriptor that indicates the end of the segment. There must be a preceding SEIStartSegment message containing the same value of segmentID. If a mismatch occurs, the decoder must ignore this message. The segment end is associated with certain layer (e.g., a group of samples, segment, sample, or sub-sample) to which this SEI descriptor is applied. The syntax of SEIStartSegmentDecriptor is as follows:

```
class SEIEndsegmentDescriptor :
        SEIMessageDescriptor(SEIEndSegmentDescriptorTag)
{
        unsigned int(32)        segmentID;
}
```

The segmentID field indicates a unique binary identifier within this stream for the segment. This value may be used to reference the segment in other SEI messages.

Storage and retrieval of audiovisual metadata has been described. Although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that any arrangement which is calculated to achieve the same purpose may be substituted for the specific embodiments shown. This application is intended to cover any adaptations or variations of the present invention.

CLAIMS

What is claimed is:

1. A method comprising:

identifying parameter set metadata defining one or more parameter sets for a plurality of portions of multimedia data; and

storing the parameter set metadata separately from the multimedia data, the separated parameter set metadata being subsequently transmitted to a decoding system for decoding the multimedia data.

2. The method of claim 1 wherein each of the plurality of portions of multimedia data is a sample within the multimedia data.

3. The method of claim 1 wherein each of the plurality of portions of multimedia data is a sub-sample within a portion of the multimedia data.

4. The method of claim 1 wherein:

the multimedia data is stored in a video track; and

the parameter set metadata is stored in a parameter track.

5. The method of claim 4 further comprising:

synchronizing the parameter track with the video track.

6. The method of claim 4 wherein the parameter track is inactive.

7. The method of claim 4 wherein each parameter set is stored in the parameter track as a parameter set sample.

8. The method of claim 1 further comprising:

transmitting the multimedia data in a video elementary stream; and

transmitting the parameter set metadata as an object descriptor stream.

9. The method of claim 8 wherein each parameter set is sent in the object descriptor stream as an object descriptor message.

10. The method of claim 8 further comprising:

synchronizing the object descriptor stream with the video elementary stream.

11. The method of claim 1 further comprising:

receiving, at the decoding system, the multimedia data and the separated parameter set metadata, the separated parameter set metadata being subsequently used to identify any of the one or more parameter sets that are required to decode at least a portion of the multimedia data.

12. A method comprising:

identifying one or more descriptions pertaining to multimedia data; and

including the one or more descriptions into supplemental enhancement information associated with the multimedia data, the SEI containing the one or more descriptions being subsequently transmitted to a decoding system for optional use in decoding of the multimedia data.

13. The method of claim 12 wherein the SEI is stored as metadata, separately from the multimedia data.

14. The method of claim 13 wherein the SEI metadata includes a plurality of SEI messages.

15. The method of claim 14 wherein each of the plurality of the SEI messages is stored as a box in a track of a movie box.

16. The method of claim 13 wherein:
the multimedia data is stored in a video track; and
the SEI metadata is stored in a SEI track.

17. The method of claim 16 further comprising:
synchronizing the SEI track with the video track.

18. The method of claim 17 wherein the SEI track contains the plurality of SEI messages in samples.

19. The method of claim 13 further comprising:
transmitting the multimedia data in a video elementary stream; and
transmitting the SEI metadata in an object content information (OCI) stream.

20. The method of claim 19 wherein each of the plurality of SEI messages is sent in the OCI stream as an OCI descriptor.

21. The method of claim 12 wherein each of one or more of descriptions is any one of a descriptor and a descriptor scheme.

22. The method of claim 14 wherein each of the plurality of SEI messages includes a payload header with data associating each of the plurality of SEI message with a corresponding portion of the multimedia data.

23. The method of claim 22 wherein the corresponding portion of the multimedia data is any one of a sample, a sub-sample, and a group of samples.

24. The method of claim 12 wherein including one or more descriptions into the SEI comprises encapsulating an MPEG-7 Systems Access Unit into one of a plurality of SEI messages.

25. The method of claim 12 further comprising:

transmitting each of the one or more descriptions in one of a plurality of SEI messages.

26. The method of claim 25 wherein each of the one or more descriptions is encoded either textually or binary.

27. An apparatus comprising:

a media file creator to form a first file containing multimedia data; and

a metadata file creator to identify parameter set metadata defining one or more parameter sets for a plurality of portions of the multimedia data, and to form a second file containing the parameter set metadata, the second file being subsequently used by a decoding system when decoding the multimedia data.

28. An apparatus comprising:

a media file creator to form a first file containing multimedia data; and

a metadata file creator to identify one or more descriptions pertaining to multimedia data, and to include the one or more descriptions into supplemental enhancement information associated with the multimedia data, the SEI containing the one or more descriptions being subsequently transmitted to a decoding system for optional use in decoding of the multimedia data.

29. An apparatus comprising:

means for identifying parameter set metadata defining one or more parameter sets for a plurality of portions of multimedia data; and

means for storing the parameter set metadata separately from the multimedia data, the separated parameter set metadata being subsequently transmitted to a decoding system for decoding the multimedia data.

30. An apparatus comprising:

means for identifying one or more descriptions pertaining to multimedia data; and

means for including the one or more descriptions into supplemental enhancement information associated with the multimedia data, the SEI containing the one or more descriptions being subsequently transmitted to a decoding system for optional use in decoding of the multimedia data.

31. A system comprising:

a memory; and

at least one processor coupled to the memory, the at least one processor executing a set of instructions which cause the at least one processor to

identify parameter set metadata defining one or more parameter sets for a plurality of portions of multimedia data, and

store the parameter set metadata separately from the multimedia data, the separated parameter set metadata being subsequently transmitted to a decoding system for decoding the multimedia data.

32. A system comprising:

a memory; and

at least one processor coupled to the memory, the at least one processor executing a set of instructions which cause the at least one processor to

identify one or more descriptions pertaining to multimedia data, and

include the one or more descriptions into supplemental enhancement information associated with the multimedia data, the SEI containing the one or more descriptions being subsequently transmitted to a decoding system for optional use in decoding of the multimedia data.

33. A computer readable medium that provides instructions, which when executed on a processor cause the processor to perform a method comprising:

identifying parameter set metadata defining one or more parameter sets for a plurality of portions of multimedia data; and

storing the parameter set metadata separately from the multimedia data, the separated parameter set metadata being subsequently transmitted to a decoding system for decoding the multimedia data.

34. A computer readable medium that provides instructions, which when executed on a processor cause the processor to perform a method comprising:

identifying one or more descriptions pertaining to multimedia data; and

including the one or more descriptions into supplemental enhancement information associated with the multimedia data, the SEI containing the one or more descriptions being subsequently transmitted to a decoding system for optional use in decoding of the multimedia data.

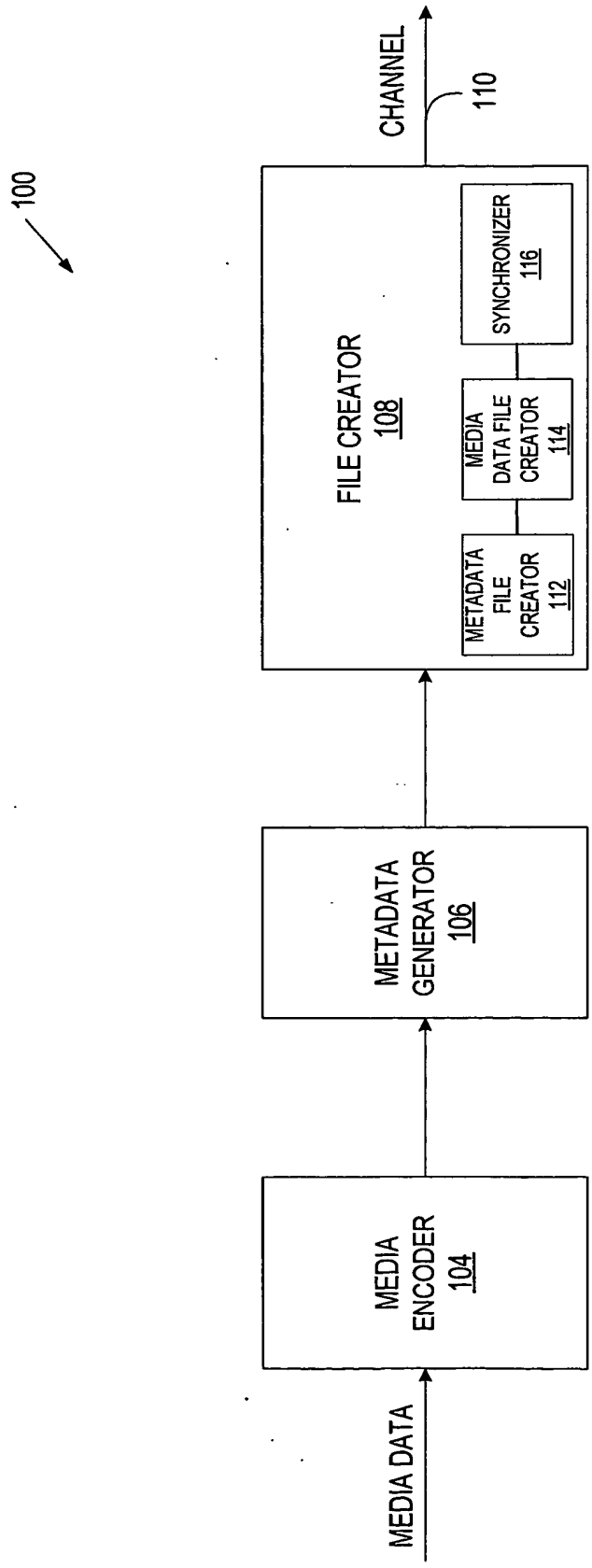

100
CHANNEL
110
FILE CREATOR
108
SYNCHRONIZER
116
MEDIA
DATA FILE
CREATOR
114
METADATA
FILE
CREATOR
112
METADATA
GENERATOR
106
MEDIA
ENCODER
104
MEDIA DATA

FIG. 1

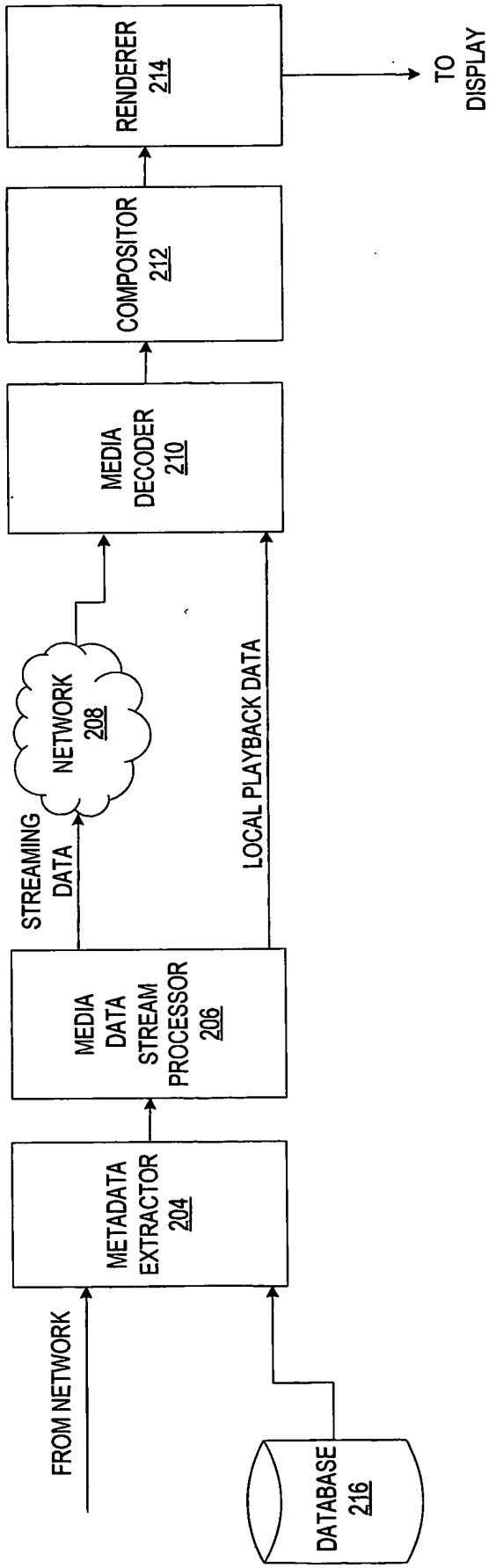

RENDERER 214
TO DISPLAY
COMPOSITOR 212
MEDIA DECODER 210
NETWORK 208
STREAMING DATA
LOCAL PLAYBACK DATA
MEDIA DATA STREAM PROCESSOR 206
METADATA EXTRACTOR 204
FROM NETWORK
DATABASE 216

FIG. 2

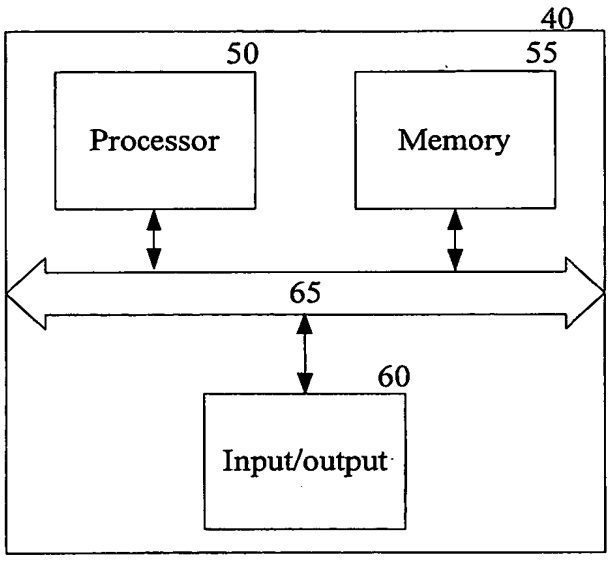


FIGURE 3

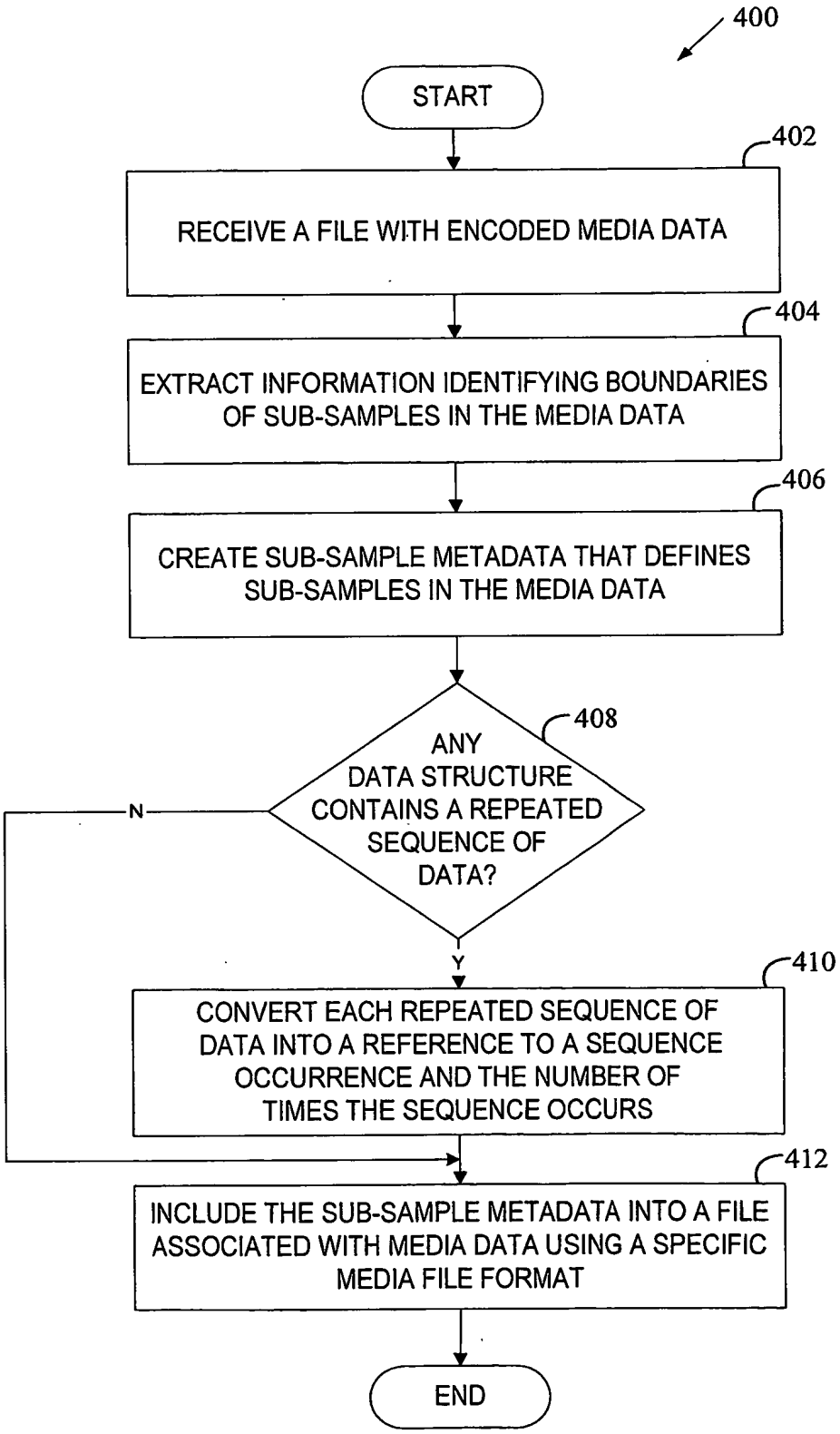

400
START
402
RECEIVE A FILE WITH ENCODED MEDIA DATA
404
EXTRACT INFORMATION IDENTIFYING BOUNDARIES OF SUB-SAMPLES IN THE MEDIA DATA
406
CREATE SUB-SAMPLE METADATA THAT DEFINES SUB-SAMPLES IN THE MEDIA DATA
408
ANY DATA STRUCTURE CONTAINS A REPEATED SEQUENCE OF DATA?
N
Y
410
CONVERT EACH REPEATED SEQUENCE OF DATA INTO A REFERENCE TO A SEQUENCE OCCURRENCE AND THE NUMBER OF TIMES THE SEQUENCE OCCURS
412
INCLUDE THE SUB-SAMPLE METADATA INTO A FILE ASSOCIATED WITH MEDIA DATA USING A SPECIFIC MEDIA FILE FORMAT
END

FIG. 4

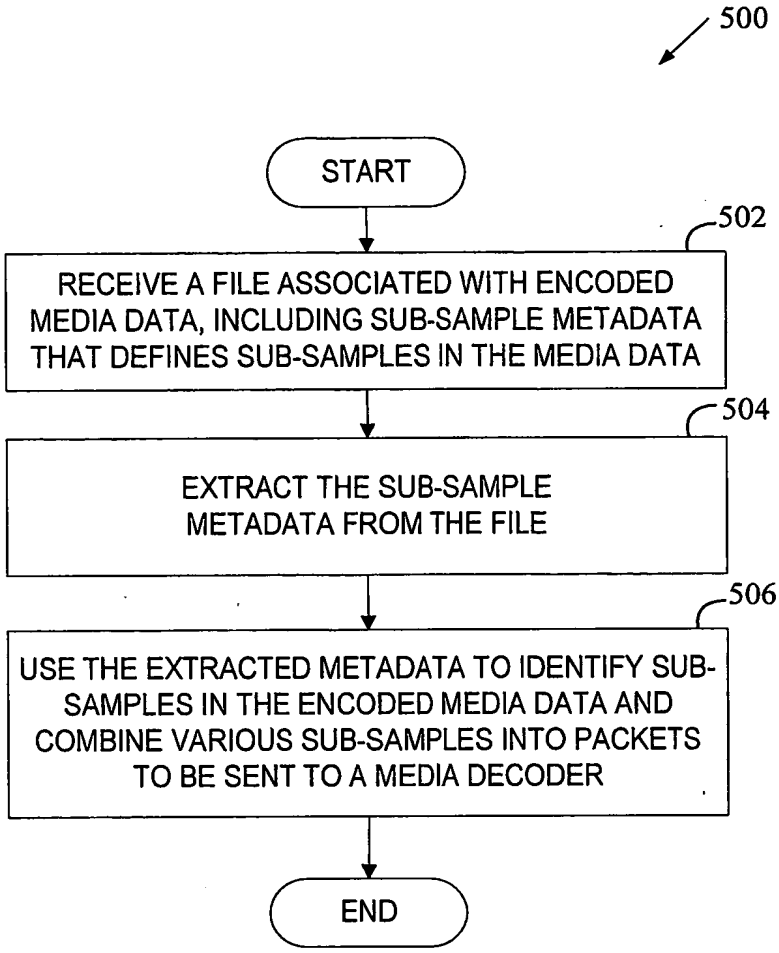

500
START
502
RECEIVE A FILE ASSOCIATED WITH ENCODED MEDIA DATA, INCLUDING SUB-SAMPLE METADATA THAT DEFINES SUB-SAMPLES IN THE MEDIA DATA
504
EXTRACT THE SUB-SAMPLE METADATA FROM THE FILE
506
USE THE EXTRACTED METADATA TO IDENTIFY SUB-SAMPLES IN THE ENCODED MEDIA DATA AND COMBINE VARIOUS SUB-SAMPLES INTO PACKETS TO BE SENT TO A MEDIA DECODER
END

FIG. 5

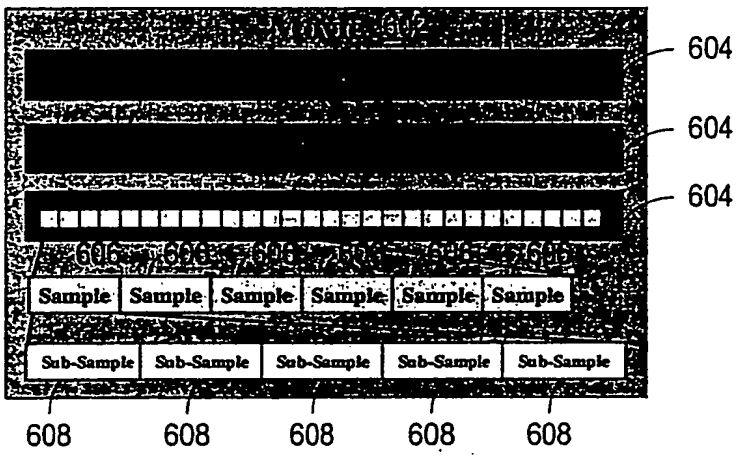

604
604
604
Sample
Sample
Sample
Sample
Sample
Sample
Sub-Sample
Sub-Sample
Sub-Sample
Sub-Sample
Sub-Sample
608
608
608
608
608

FIG. 6

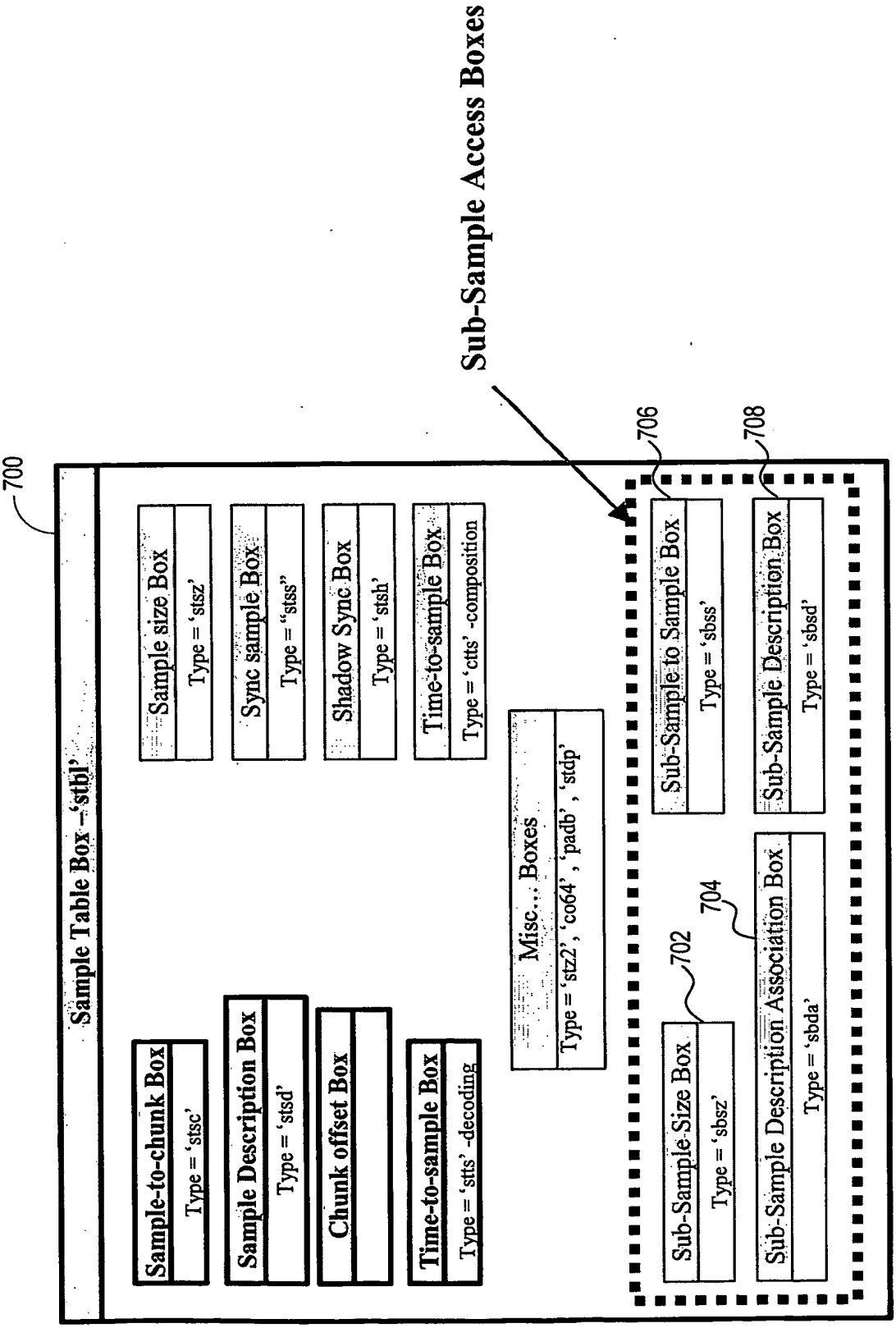

Sub-Sample Access Boxes
700
Sample Table Box – 'stbl'
Sample-to-chunk Box
Type = 'stsc'
Sample Description Box
Type = 'stsd'
Chunk offset Box
Time-to-sample Box
Type = 'stts' -decoding
Sample size Box
Type = 'stsz'
Sync sample Box
Type = "stss"
Shadow Sync Box
Type = 'stsh'
Time-to-sample Box
Type = 'ctts' -composition
Misc... Boxes
Type = 'stz2', 'co64', 'padb', 'stdp'
706
708
Sub-Sample to Sample Box
Type = 'sbss'
Sub-Sample Description Box
Type = 'sbsd'
Sub-Sample Size Box
Type = 'sbsz'
702
704
Sub-Sample Description Association Box
Type = 'sbda'

FIG. 7A

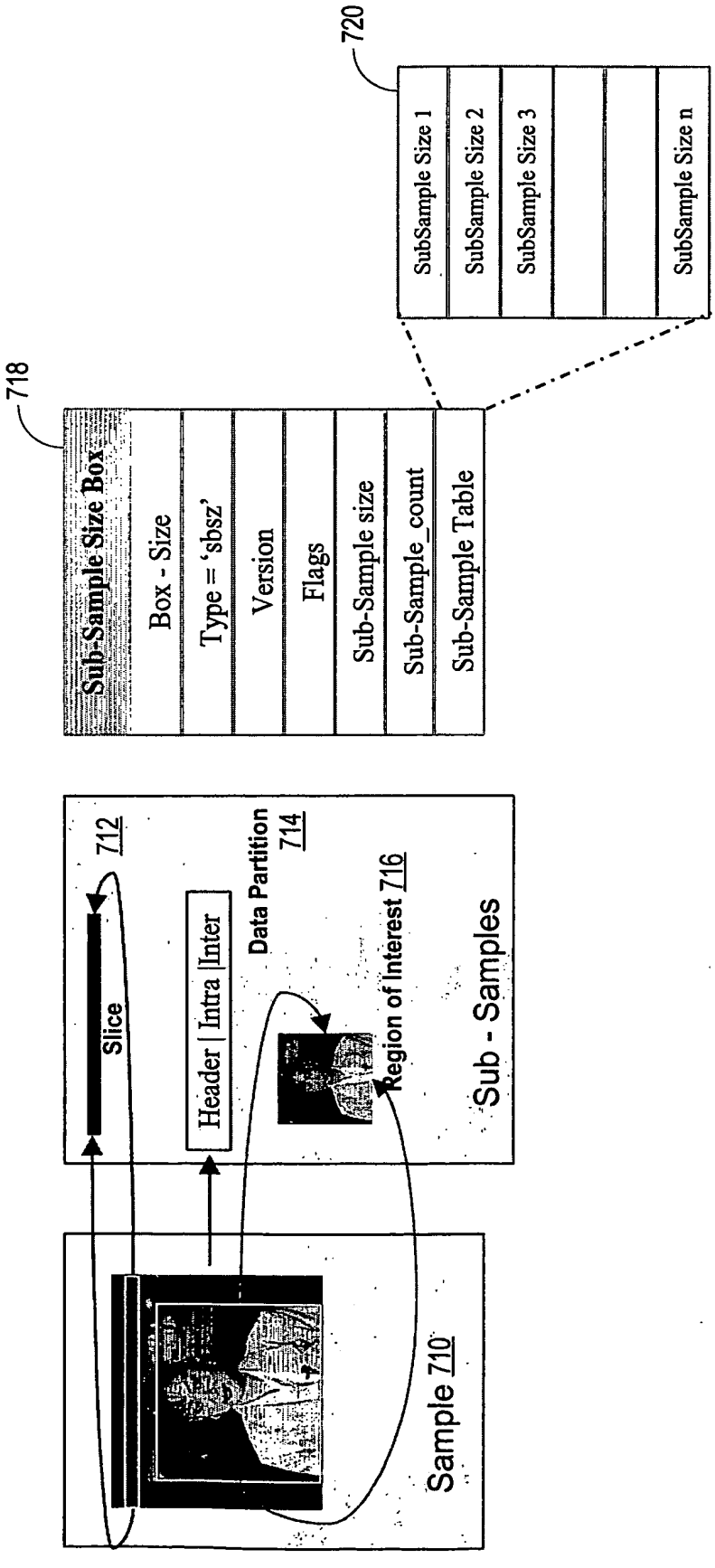

720
SubSample Size 1
SubSample Size 2
SubSample Size 3
SubSample Size n
718
Sub-Sample Size Box
Box - Size
Type = 'sbsz'
Version
Flags
Sub-Sample size
Sub-Sample_count
Sub-Sample Table
712
Slice
Header | Intra | Inter
Data Partition 714
Region of Interest 716
Sub - Samples
Sample 710

FIG. 7B

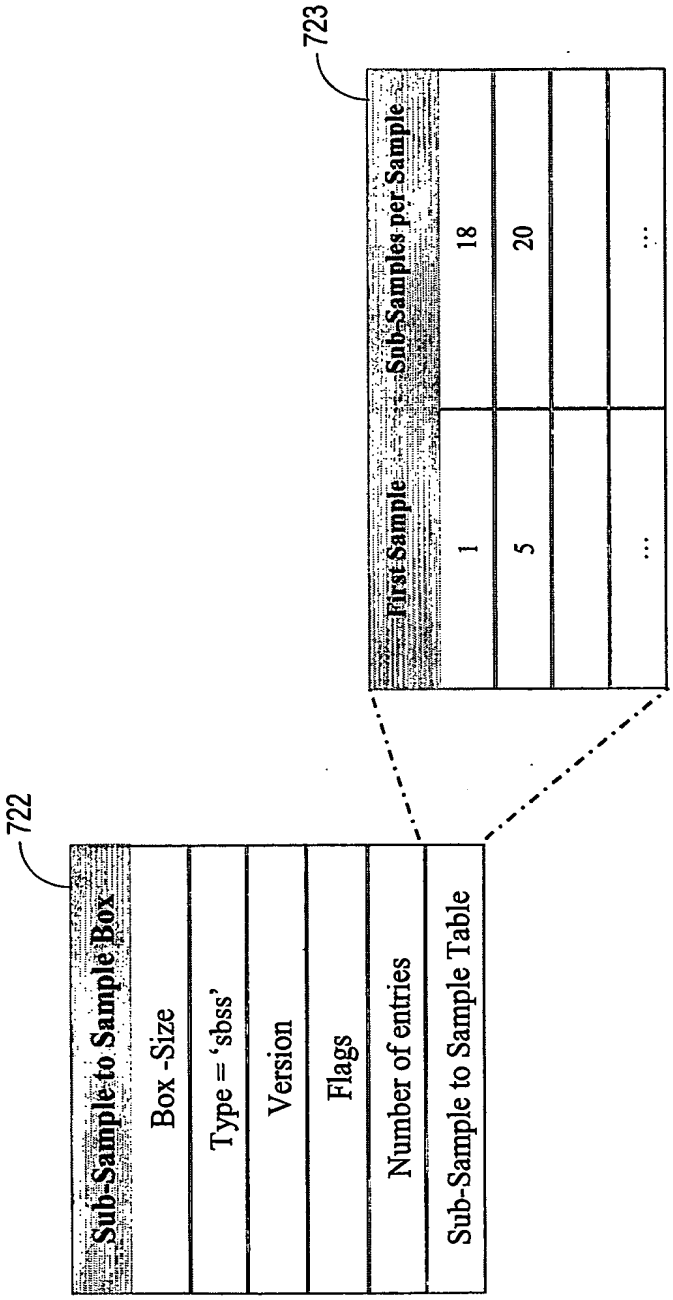

722
Sub-Sample to Sample Box
Box -Size
Type = 'sbss'
Version
Flags
Number of entries
Sub-Sample to Sample Table
723
First Sample
Sub-Samples per Sample
1
18
5
20
...
...

FIG. 7C

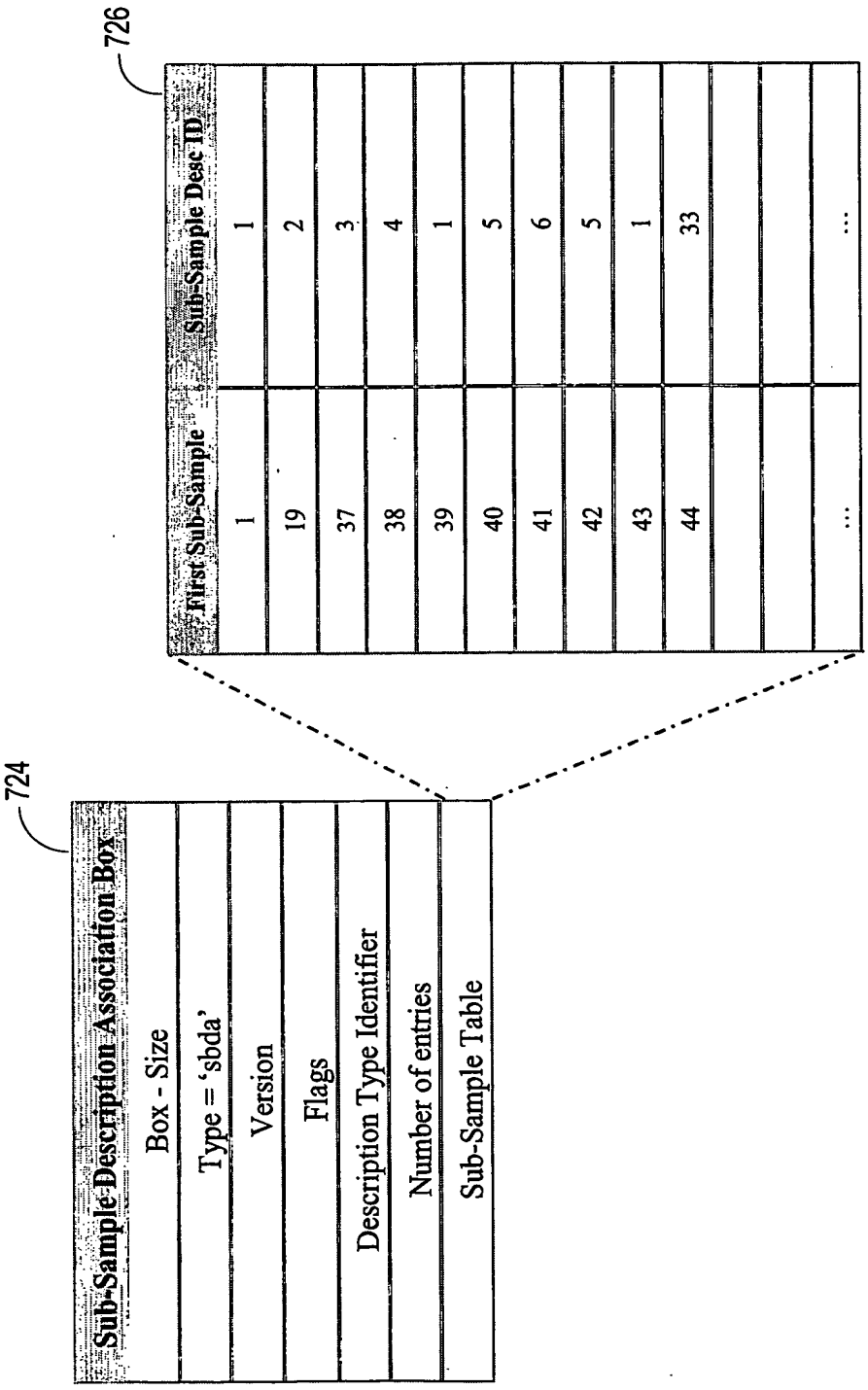

724
Sub-Sample Description Association Box
Box - Size
Type = 'sbda'
Version
Flags
Description Type Identifier
Number of entries
Sub-Sample Table
726
First Sub-Sample
Sub-Sample Desc ID
1 1
19 2
37 3
38 4
39 1
40 5
41 6
42 5
43 1
44 33
... ...

FIG. 7D

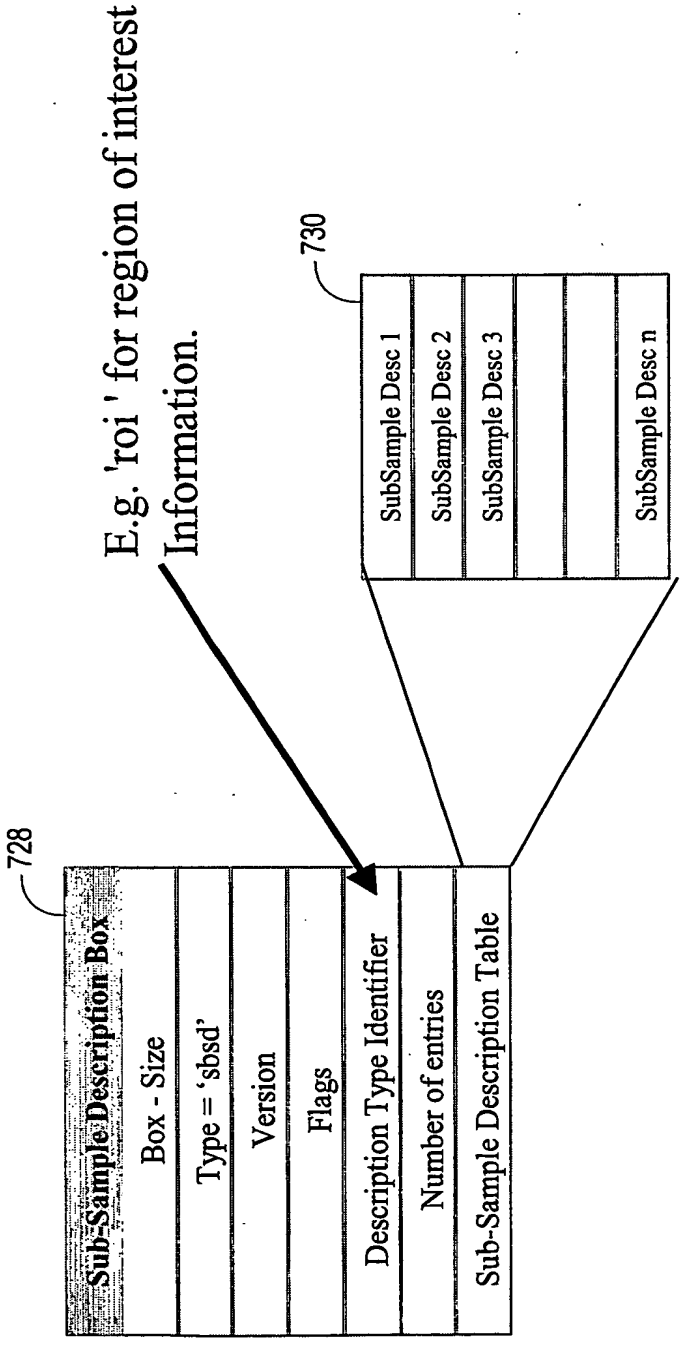

728
Sub-Sample Description Box
Box - Size
Type = 'sbsd'
Version
Flags
Description Type Identifier
Number of entries
Sub-Sample Description Table
E.g. 'roi ' for region of interest Information.
730
SubSample Desc 1
SubSample Desc 2
SubSample Desc 3
SubSample Desc n

FIG. 7E

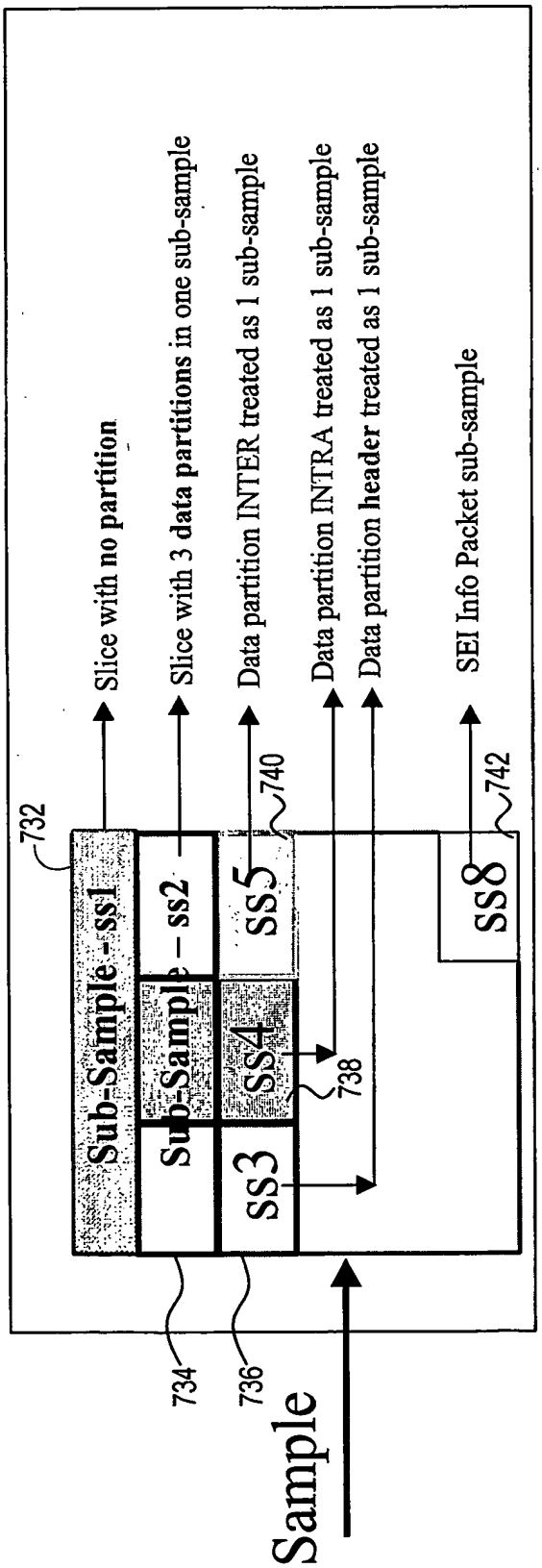

Sample
734
736
732
Sub-Sample - ss1
Sub-Sample – ss2
ss3
ss4
ss5
ss8
738
740
742
Slice with no partition
Slice with 3 data partitions in one sub-sample
Data partition INTER treated as 1 sub-sample
Data partition INTRA treated as 1 sub-sample
Data partition header treated as 1 sub-sample
SEI Info Packet sub-sample

FIG. 7F

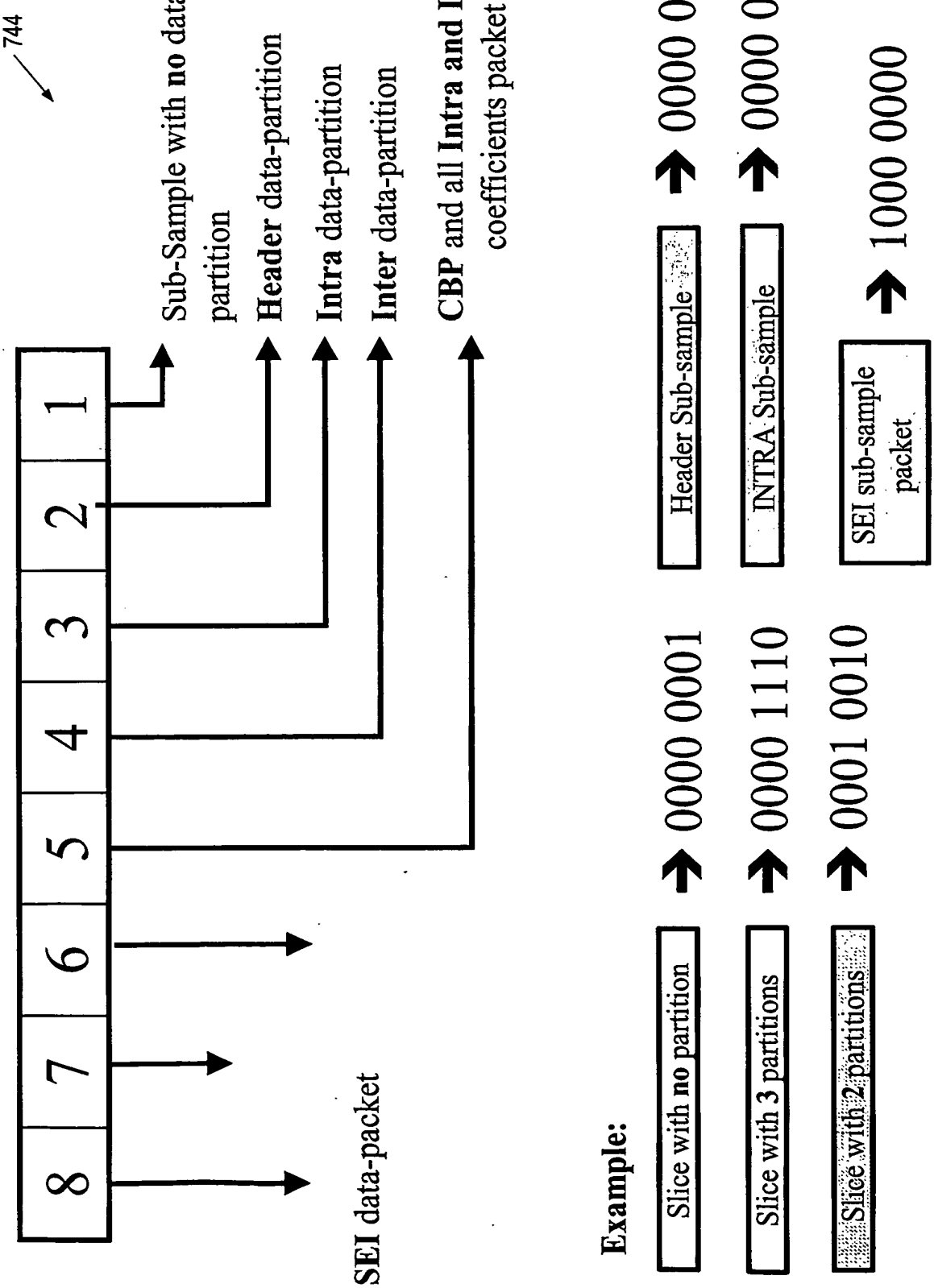

744
8
7
6
5
4
3
2
1
SEI data-packet
CBP and all Intra and Inter coefficients packet
Inter data-partition
Intra data-partition
Header data-partition
Sub-Sample with no data-partition
Example:
Slice with no partition
0000 0001
Header Sub-sample
0000 0010
Slice with 3 partitions
0000 1110
INTRA Sub-sample
0000 0100
Slice with 2 partitions
0001 0010
SEI sub-sample packet
1000 0000

FIG. 7G

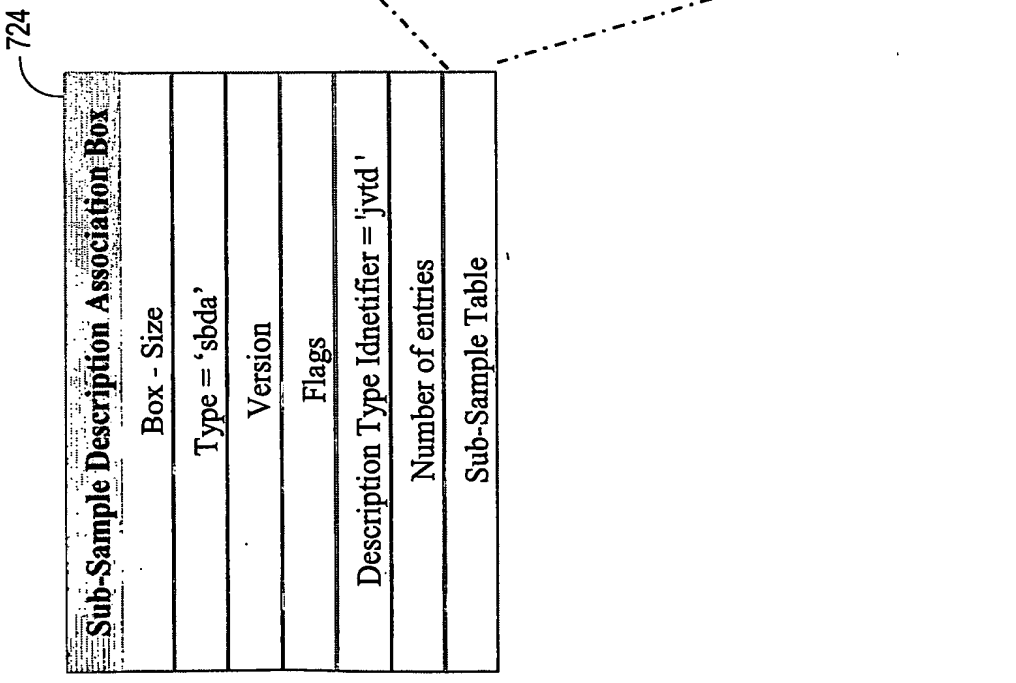

724
Sub-Sample Description Association Box
Box - Size
Type = 'sbda'
Version
Flags
Description Type Idnetifier = 'jvtd'
Number of entries
Sub-Sample Table
726
32 bits
First Sub-Sample
Sub-Sample Desc ID
1
19
37
38
39
40
41
42
43
44
... 0000 0001
... 0000 1110
... 0001 0010
... 1000 0000
... 0000 0010
... 0000 0100
... 0000 1000
... 0000 0010
... 0000 0100
... 0000 1000
no data -partition
data -partitioned slices
SEI packet
Header packet
INTRA packet
INTER packet
Header packet
INTRA packet
INTER packet

FIG. 7H

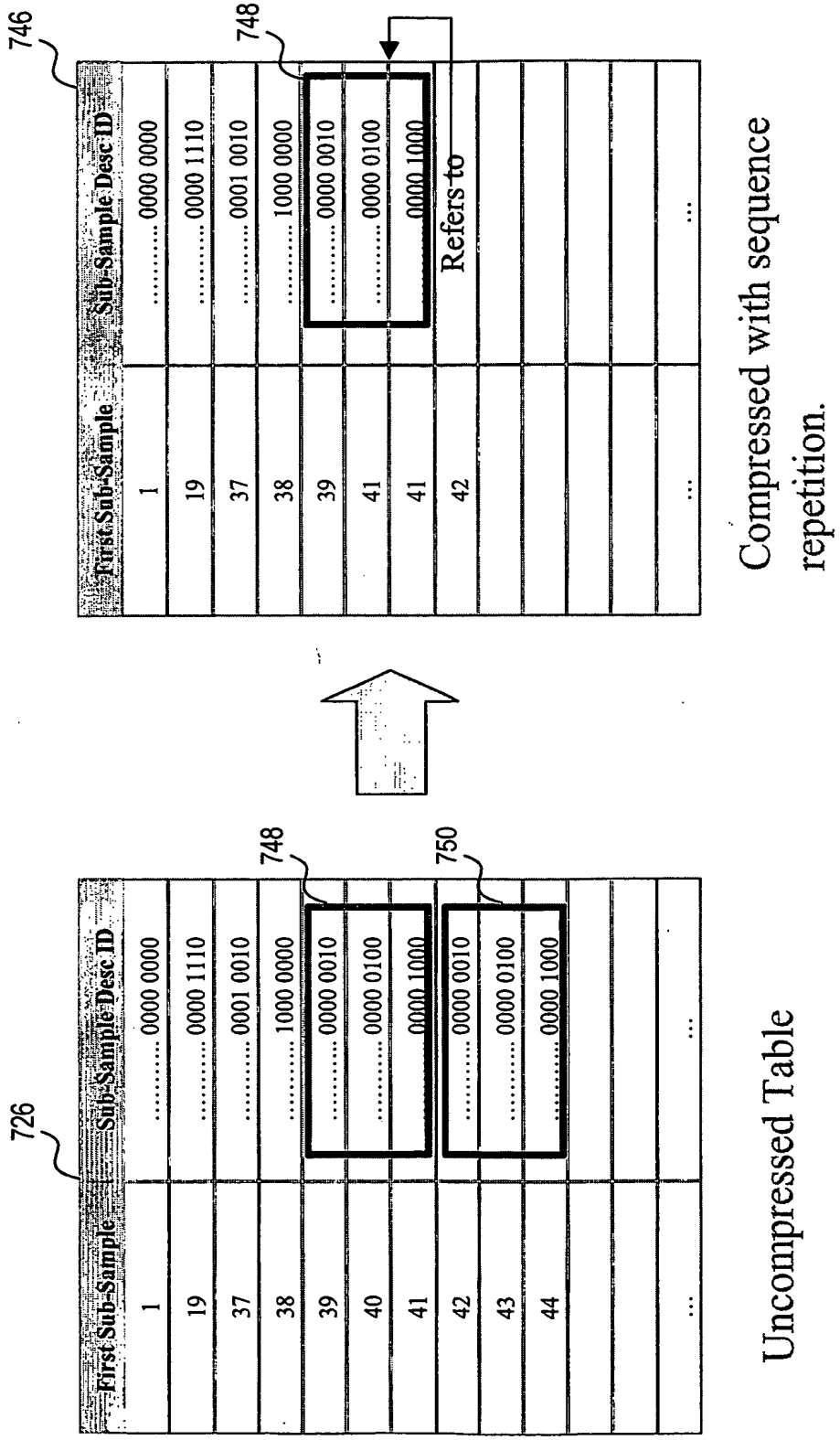

726
First Sub-Sample
Sub-Sample Desc ID
1
19
37
38
39
40
41
42
43
44
748
750
0000 0000
0000 1110
0001 0010
1000 0000
0000 0010
0000 0100
0000 1000
746
748
Refers to
Uncompressed Table
Compressed with sequence repetition.

FIG. 7I

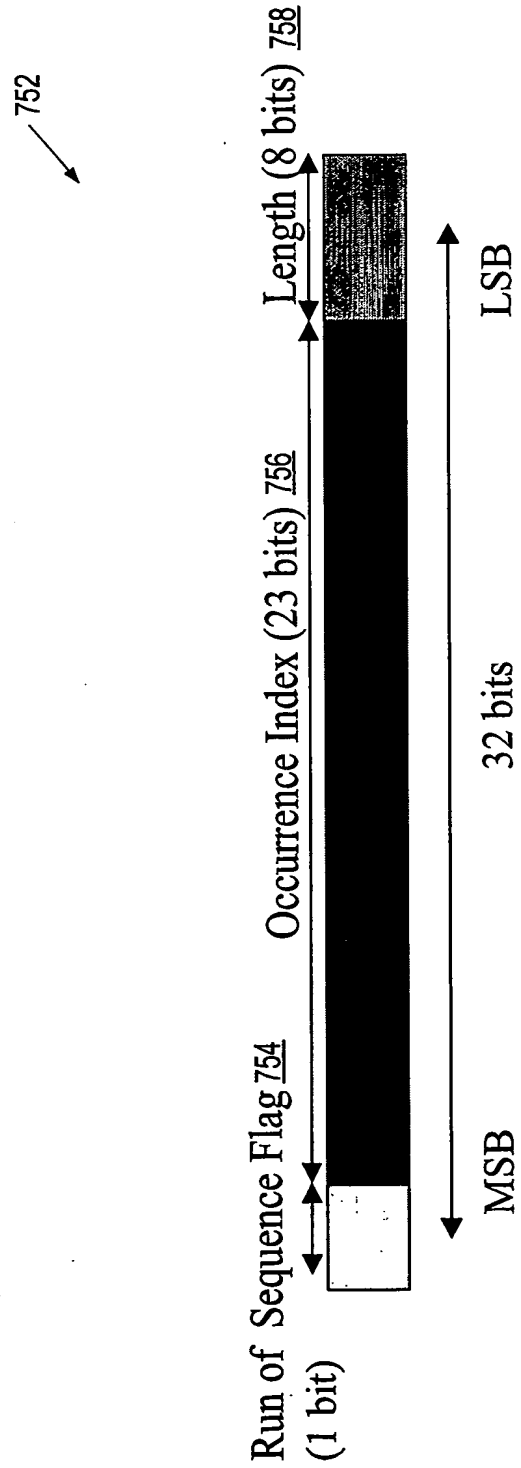
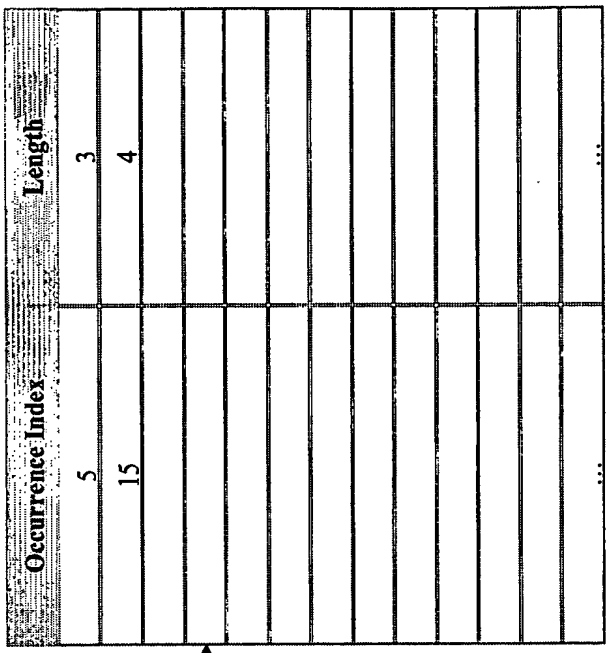
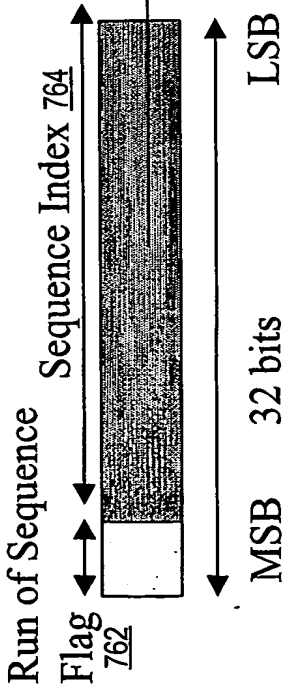

752
Length (8 bits) 758
Occurrence Index (23 bits) 756
Run of Sequence Flag 754
(1 bit)
LSB
32 bits
MSB

FIG. 7J



Repeated Sequence Occurrence Table 760



Structure of a Description ID field.

FIG. 7K

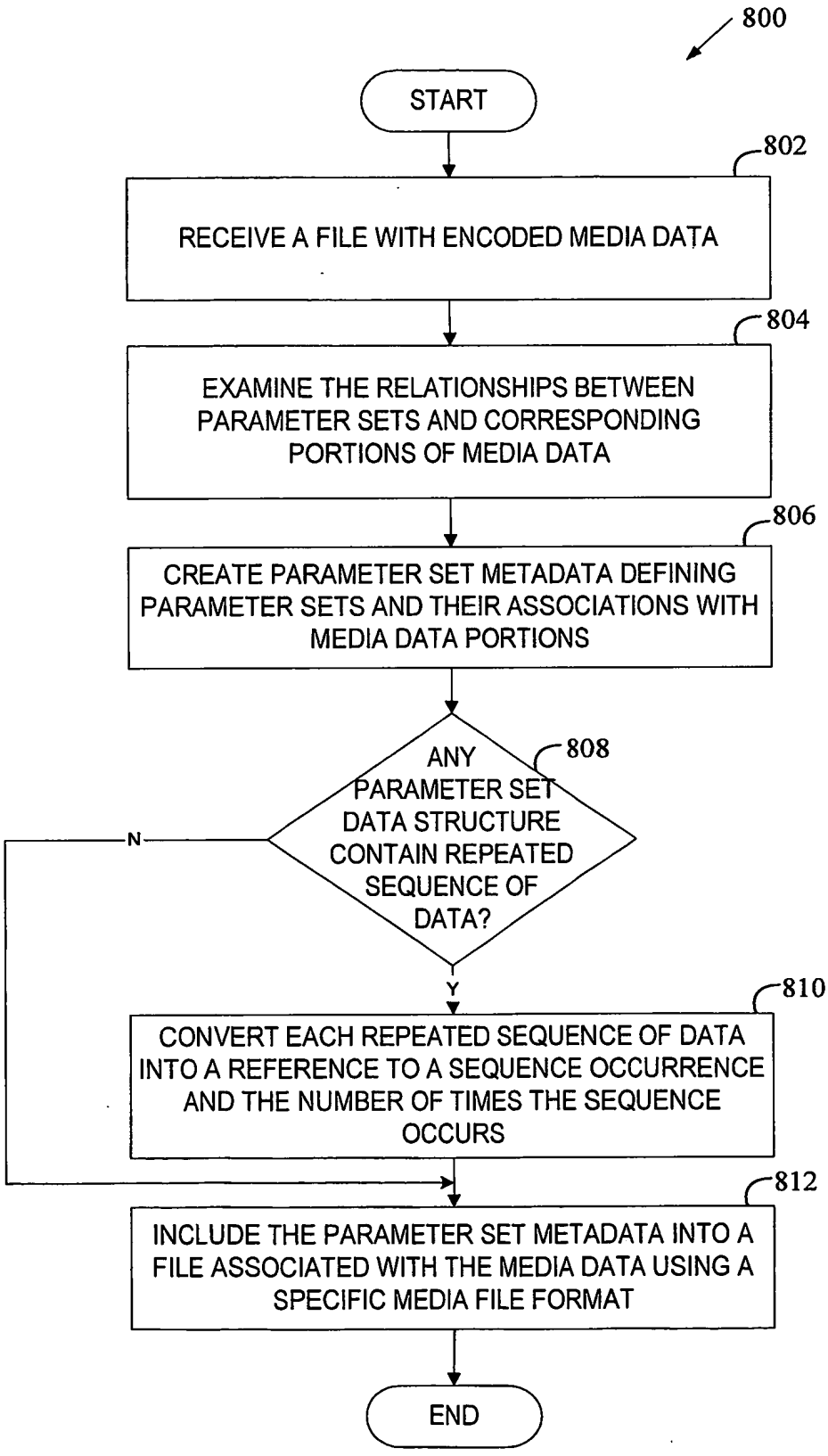

800
START
802
RECEIVE A FILE WITH ENCODED MEDIA DATA
804
EXAMINE THE RELATIONSHIPS BETWEEN PARAMETER SETS AND CORRESPONDING PORTIONS OF MEDIA DATA
806
CREATE PARAMETER SET METADATA DEFINING PARAMETER SETS AND THEIR ASSOCIATIONS WITH MEDIA DATA PORTIONS
808
ANY PARAMETER SET DATA STRUCTURE CONTAIN REPEATED SEQUENCE OF DATA?
N
Y
810
CONVERT EACH REPEATED SEQUENCE OF DATA INTO A REFERENCE TO A SEQUENCE OCCURRENCE AND THE NUMBER OF TIMES THE SEQUENCE OCCURS
812
INCLUDE THE PARAMETER SET METADATA INTO A FILE ASSOCIATED WITH THE MEDIA DATA USING A SPECIFIC MEDIA FILE FORMAT
END

FIG. 8

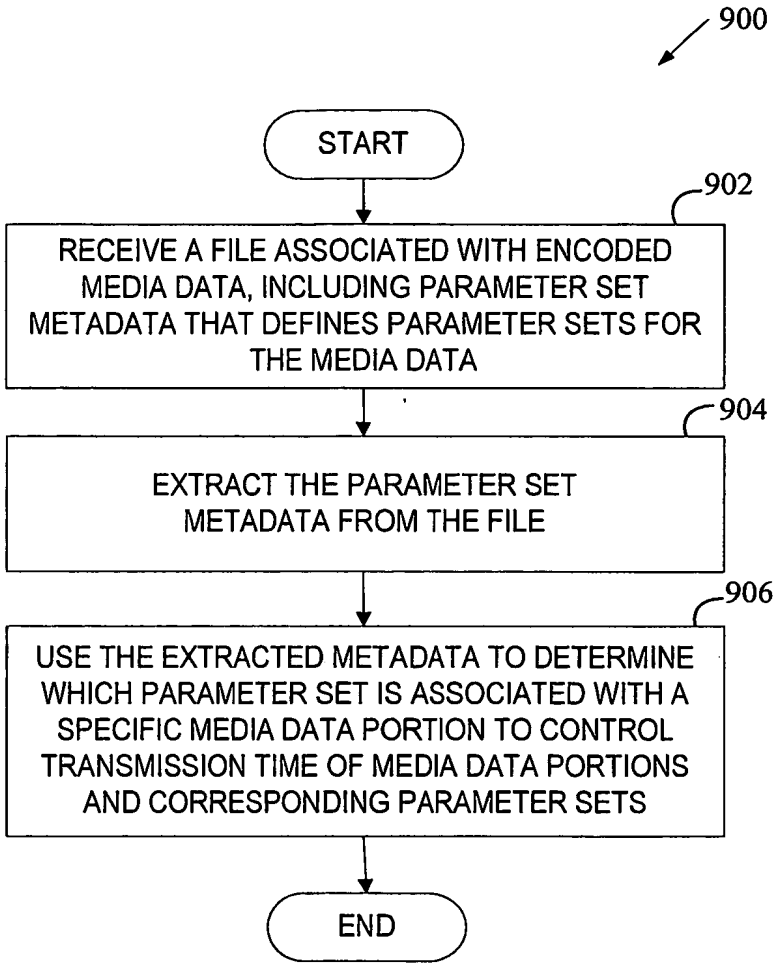

900
START
902
RECEIVE A FILE ASSOCIATED WITH ENCODED MEDIA DATA, INCLUDING PARAMETER SET METADATA THAT DEFINES PARAMETER SETS FOR THE MEDIA DATA
904
EXTRACT THE PARAMETER SET METADATA FROM THE FILE
906
USE THE EXTRACTED METADATA TO DETERMINE WHICH PARAMETER SET IS ASSOCIATED WITH A SPECIFIC MEDIA DATA PORTION TO CONTROL TRANSMISSION TIME OF MEDIA DATA PORTIONS AND CORRESPONDING PARAMETER SETS
END

FIG. 9

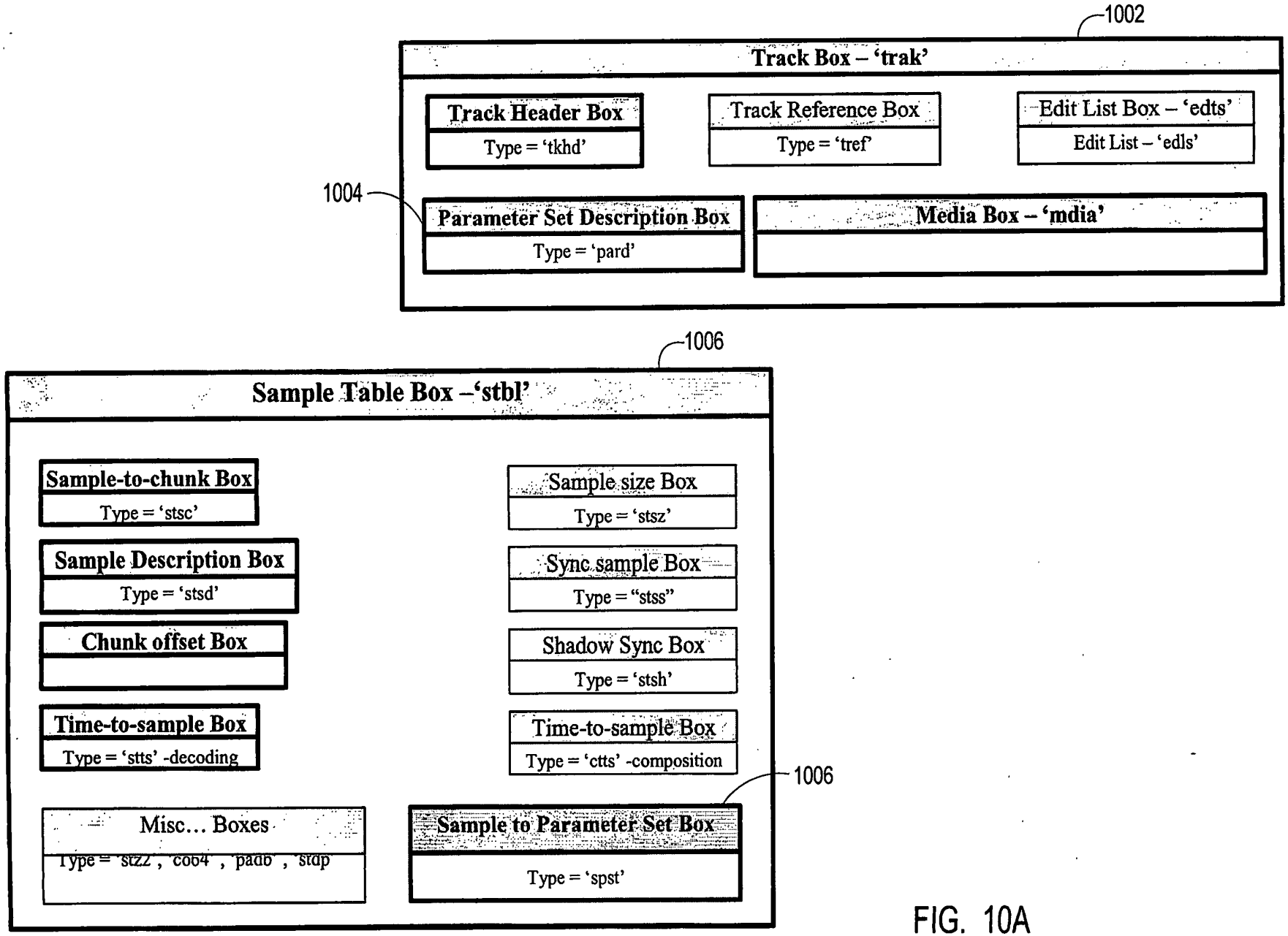

1002
Track Box – 'trak'
Track Header Box
Type = 'tkhd'
Track Reference Box
Type = 'tref'
Edit List Box – 'edts'
Edit List – 'edls'
1004
Parameter Set Description Box
Type = 'pard'
Media Box – 'mdia'
1006
Sample Table Box –'stbl'
Sample-to-chunk Box
Type = 'stsc'
Sample size Box
Type = 'stsz'
Sample Description Box
Type = 'stsd'
Sync sample Box
Type = "stss"
Chunk offset Box
Shadow Sync Box
Type = 'stsh'
Time-to-sample Box
Type = 'stts' -decoding
Time-to-sample Box
Type = 'ctts' -composition
1006
Misc... Boxes
Type = 'stzz', 'co64', 'padb', 'stdp
Sample to Parameter Set Box
Type = 'spst'

FIG. 10A

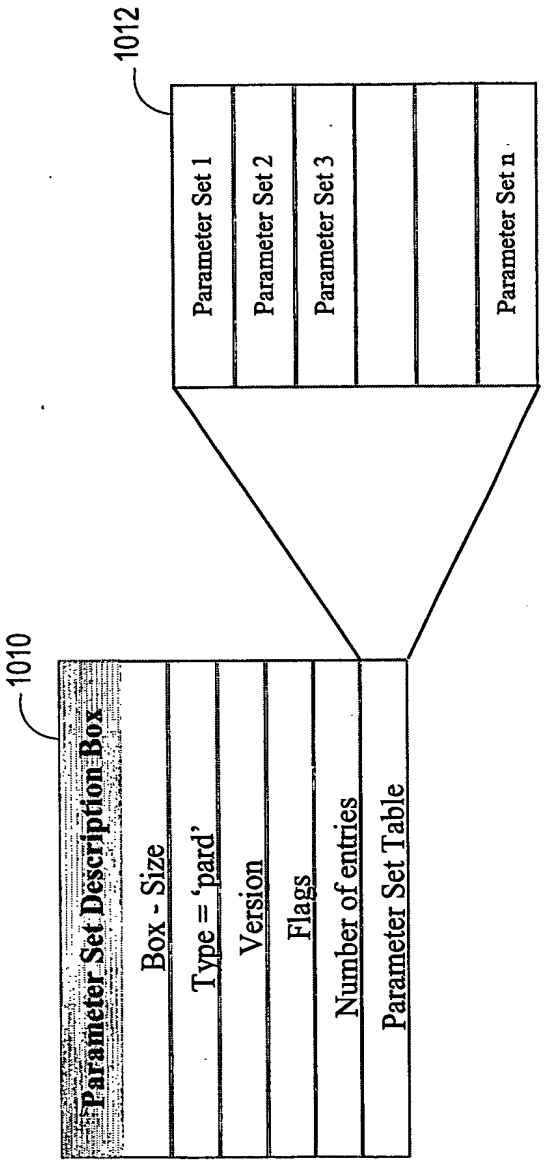

1012
Parameter Set 1
Parameter Set 2
Parameter Set 3
Parameter Set n
1010
Parameter Set Description Box
Box - Size
Type = 'pard'
Version
Flags
Number of entries
Parameter Set Table

FIG. 10B

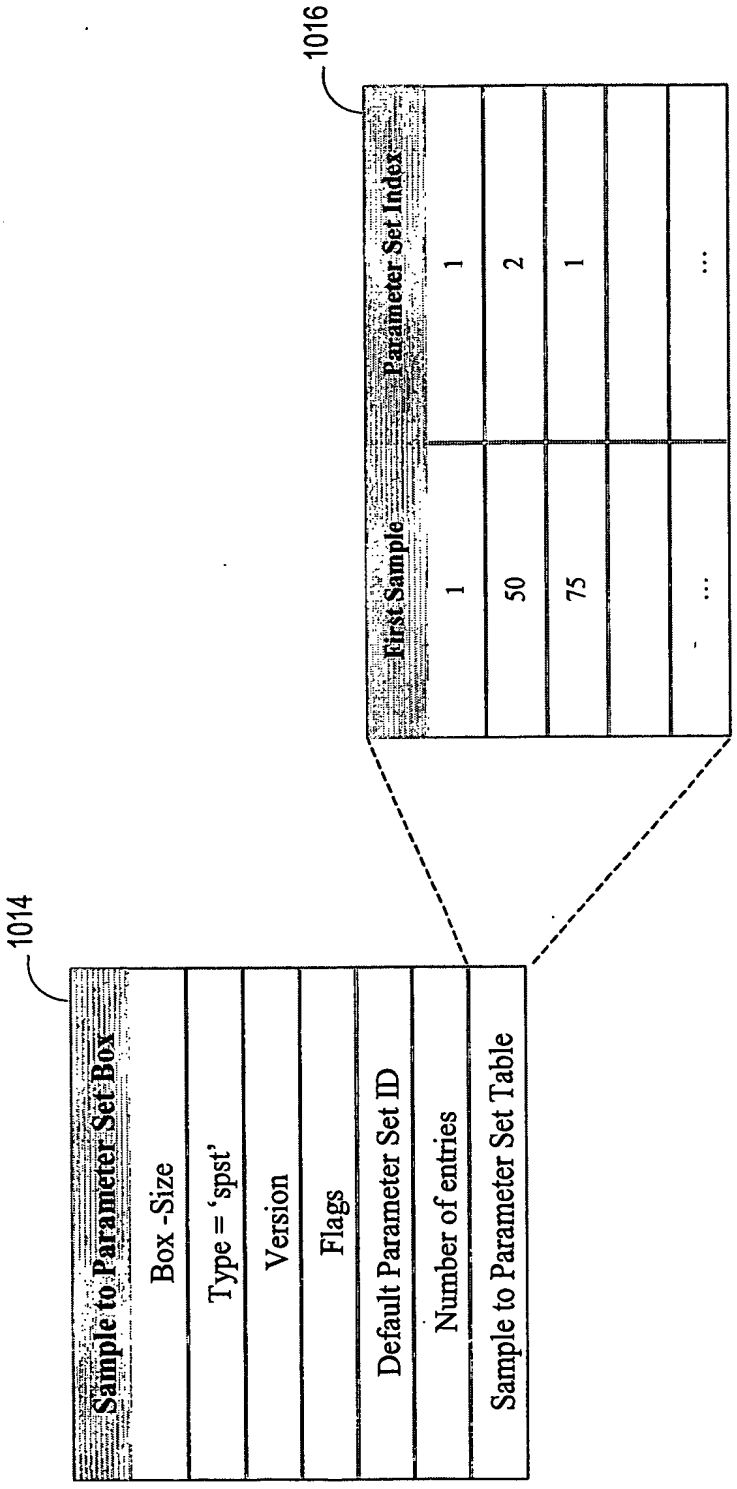

1014
Sample to Parameter Set Box
Box -Size
Type = 'spst'
Version
Flags
Default Parameter Set ID
Number of entries
Sample to Parameter Set Table
1016
First Sample
Parameter Set Index
1
50
75
...
1
2
1
...

FIG. 10C

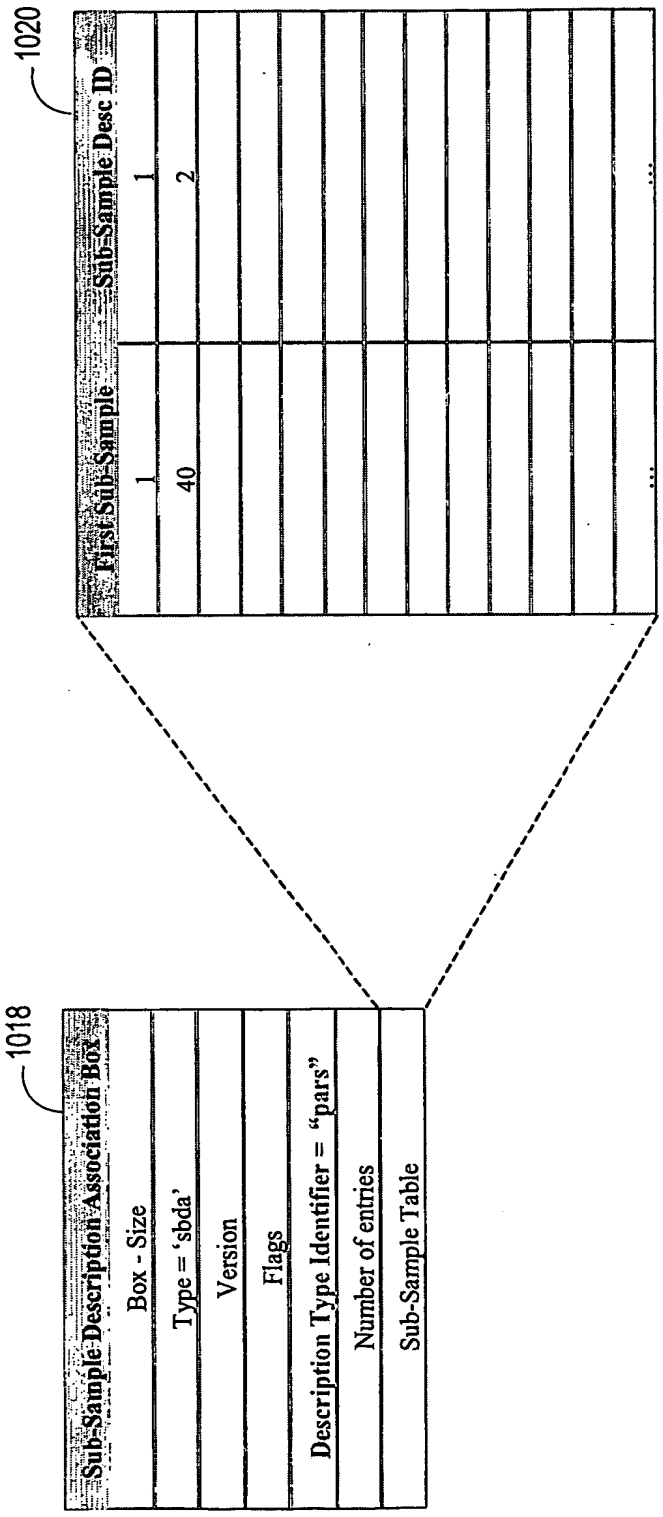

1018
Sub-Sample Description Association Box
Box - Size
Type = 'sbda'
Version
Flags
Description Type Identifier = "pars"
Number of entries
Sub-Sample Table
1020
First Sub-Sample
Sub-Sample Desc ID
1
40
1
2
...
...

FIG. 10D

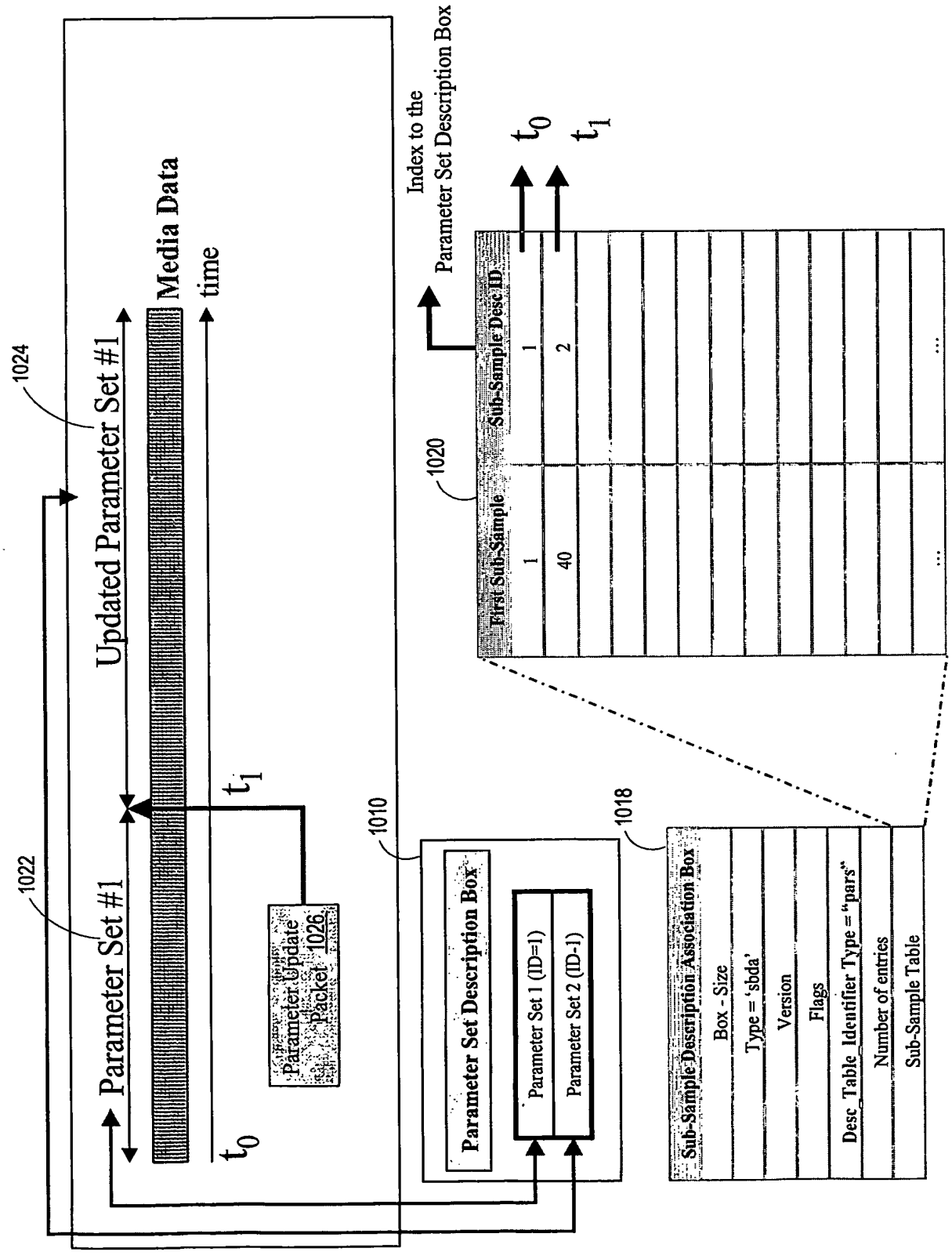

1024
Updated Parameter Set #1
Media Data
time
1022
Parameter Set #1
t_0
t_1
Parameter Update Packet 1026
1010
Parameter Set Description Box
Parameter Set 1 (ID=1)
Parameter Set 2 (ID-1)
1018
Sub-Sample Description Association Box
Box - Size
Type = 'sbda'
Version
Flags
Desc Table Identifier Type = "pars"
Number of entries
Sub-Sample Table
Index to the Parameter Set Description Box
1020
First Sub-Sample
Sub-Sample Desc ID
1
40
1
2
...
...
t_0
t_1

FIG. 10E

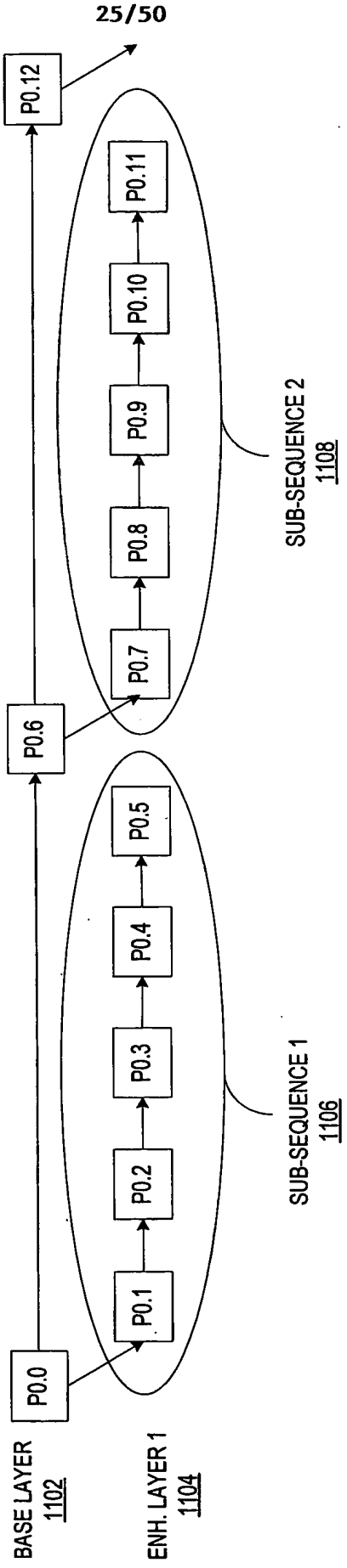

P0.12
P0.11
P0.10
P0.9
P0.8
P0.7
P0.6
P0.5
P0.4
P0.3
P0.2
P0.1
P0.0
SUB-SEQUENCE 2
1108
SUB-SEQUENCE 1
1106
BASE LAYER
1102
ENH. LAYER 1
1104

FIG. 11

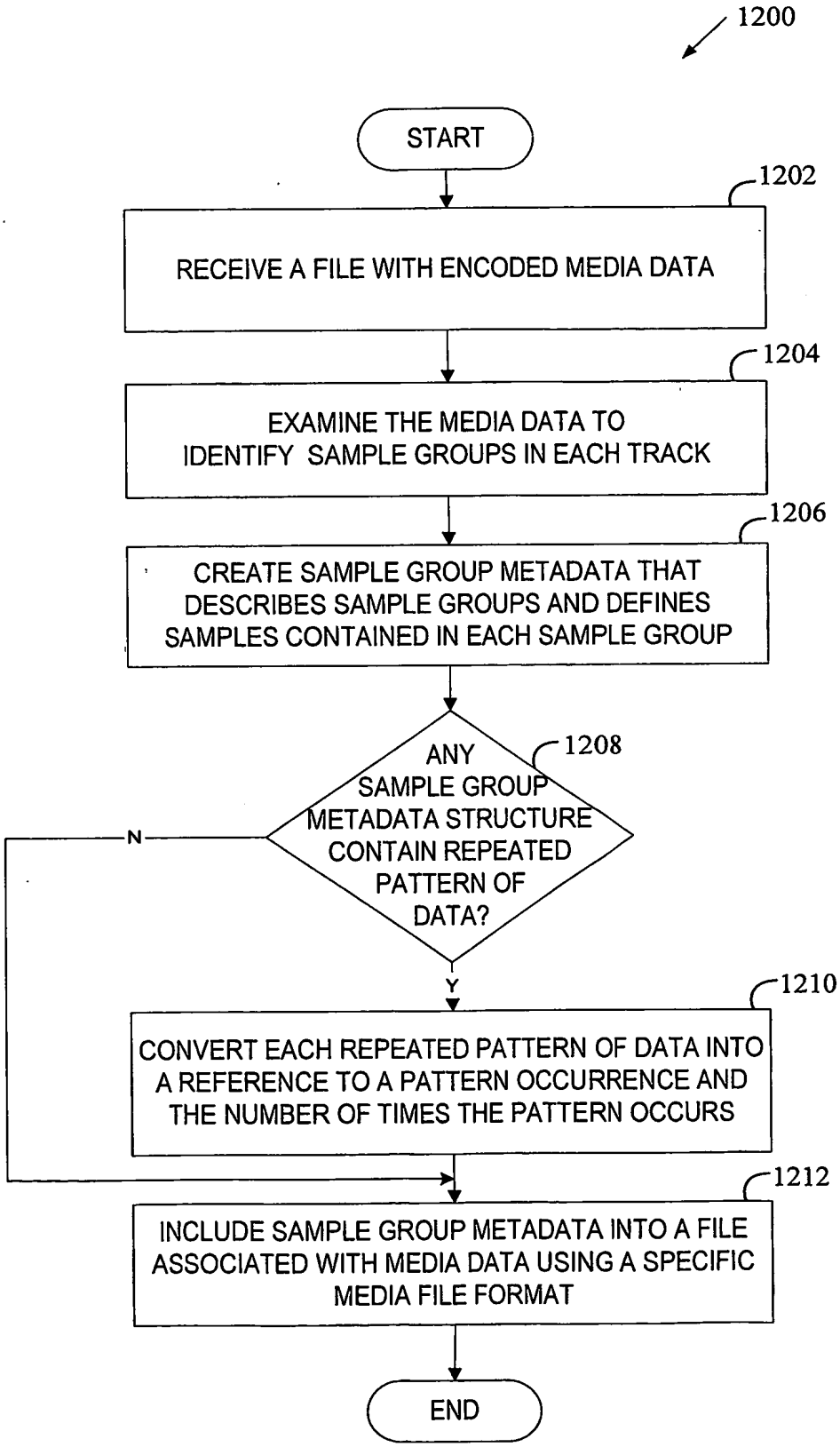

1200
START
1202
RECEIVE A FILE WITH ENCODED MEDIA DATA
1204
EXAMINE THE MEDIA DATA TO IDENTIFY SAMPLE GROUPS IN EACH TRACK
1206
CREATE SAMPLE GROUP METADATA THAT DESCRIBES SAMPLE GROUPS AND DEFINES SAMPLES CONTAINED IN EACH SAMPLE GROUP
1208
ANY SAMPLE GROUP METADATA STRUCTURE CONTAIN REPEATED PATTERN OF DATA?
N
Y
1210
CONVERT EACH REPEATED PATTERN OF DATA INTO A REFERENCE TO A PATTERN OCCURRENCE AND THE NUMBER OF TIMES THE PATTERN OCCURS
1212
INCLUDE SAMPLE GROUP METADATA INTO A FILE ASSOCIATED WITH MEDIA DATA USING A SPECIFIC MEDIA FILE FORMAT
END

FIG. 12

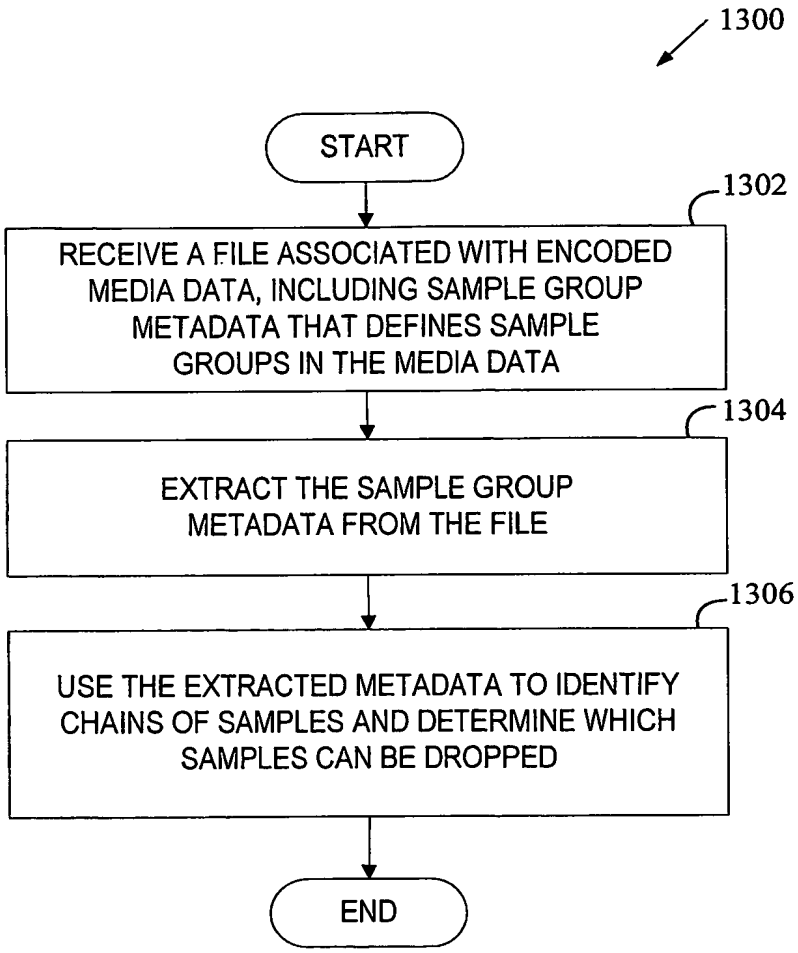

1300
START
1302
RECEIVE A FILE ASSOCIATED WITH ENCODED MEDIA DATA, INCLUDING SAMPLE GROUP METADATA THAT DEFINES SAMPLE GROUPS IN THE MEDIA DATA
1304
EXTRACT THE SAMPLE GROUP METADATA FROM THE FILE
1306
USE THE EXTRACTED METADATA TO IDENTIFY CHAINS OF SAMPLES AND DETERMINE WHICH SAMPLES CAN BE DROPPED
END

FIG. 13

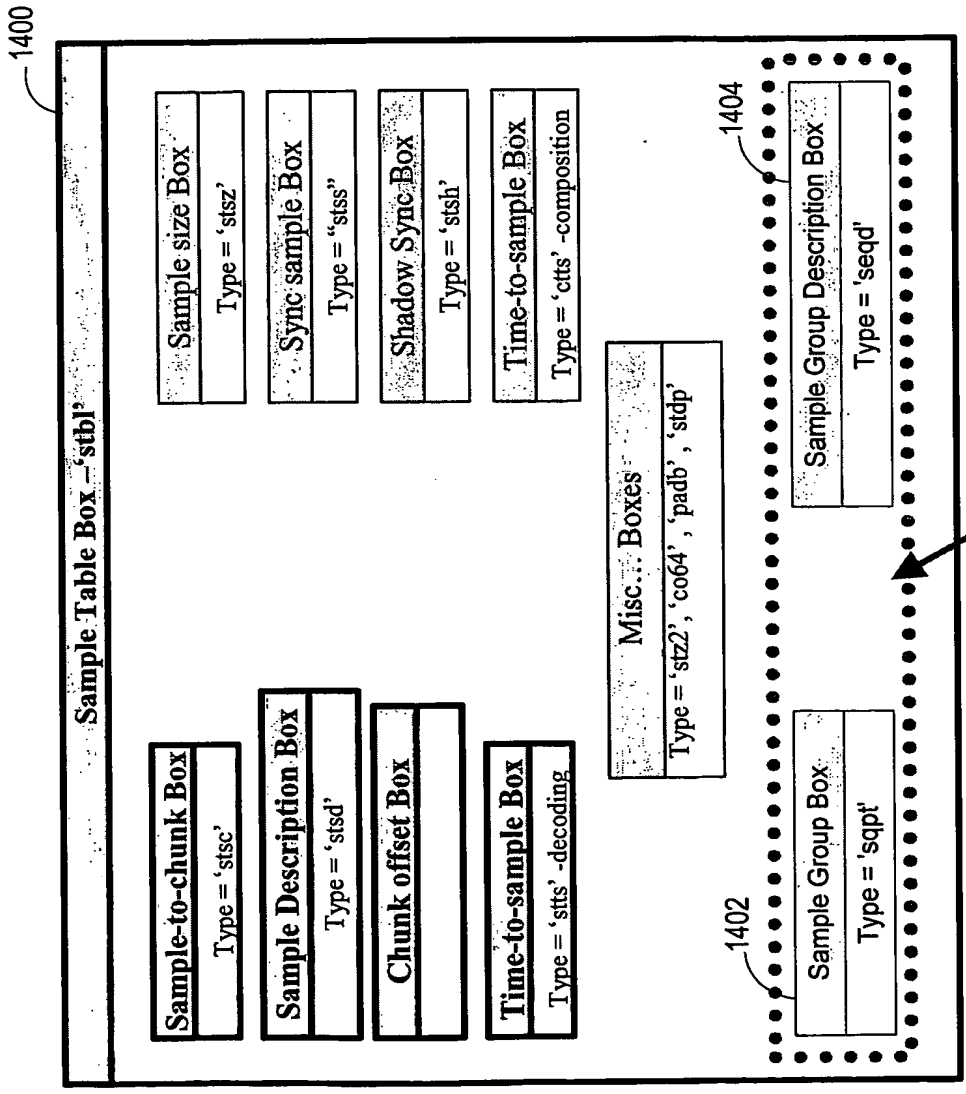

1400
Sample Table Box – 'stbl'
Sample-to-chunk Box
Type = 'stsc'
Sample Description Box
Type = 'stsd'
Chunk offset Box
Time-to-sample Box
Type = 'stts' -decoding
Sample size Box
Type = 'stsz'
Sync sample Box
Type = "stss"
Shadow Sync Box
Type = 'stsh'
Time-to-sample Box
Type = 'ctts' -composition
Misc... Boxes
Type = 'stz2', 'co64', 'padb', 'stdp'
1402
Sample Group Box
Type = 'sqpt'
1404
Sample Group Description Box
Type = 'seqd'
SAMPLE GROUPS

FIG. 14A

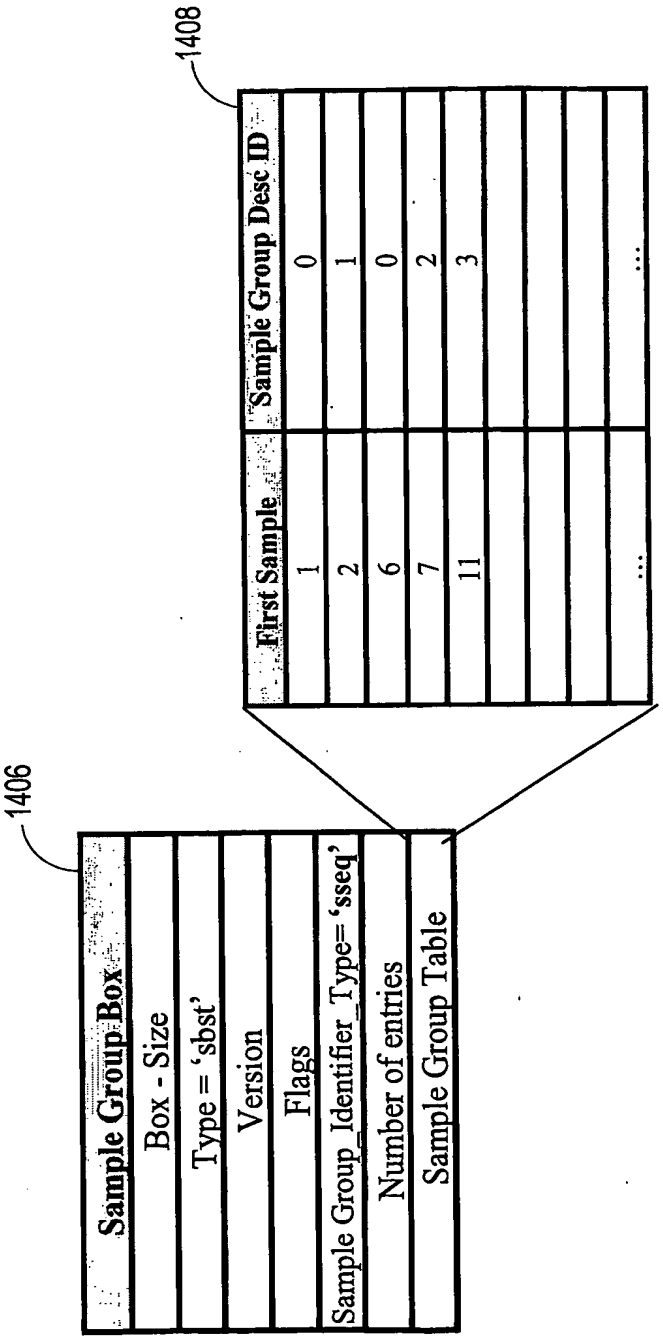

1406
Sample Group Box
Box - Size
Type = 'sbst'
Version
Flags
Sample Group_Identifier_Type= 'sseq'
Number of entries
Sample Group Table
1408
First Sample
Sample Group Desc ID
1
0
2
1
6
0
7
2
11
3
...
...

FIG. 14B

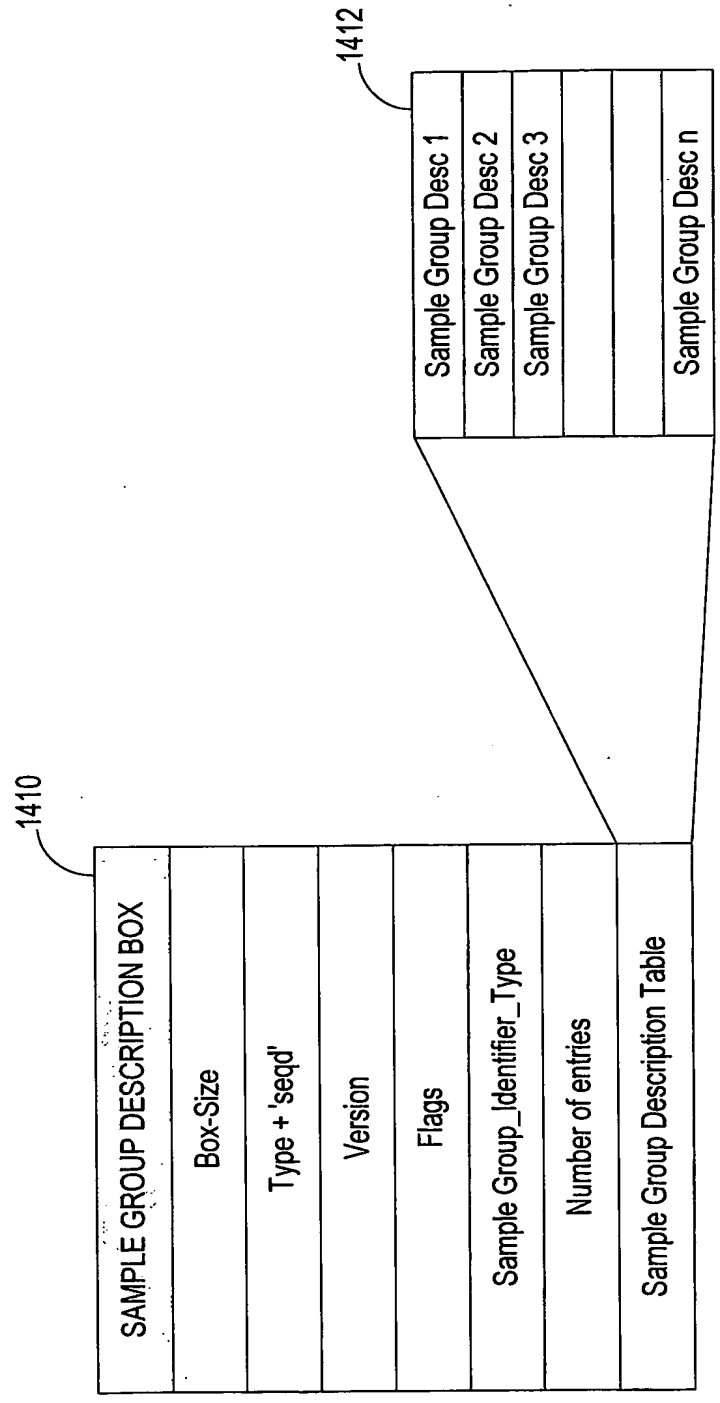

1410
SAMPLE GROUP DESCRIPTION BOX
Box-Size
Type + 'seqd'
Version
Flags
Sample Group_Identifier_Type
Number of entries
Sample Group Description Table
1412
Sample Group Desc 1
Sample Group Desc 2
Sample Group Desc 3
Sample Group Desc n

FIG. 14C

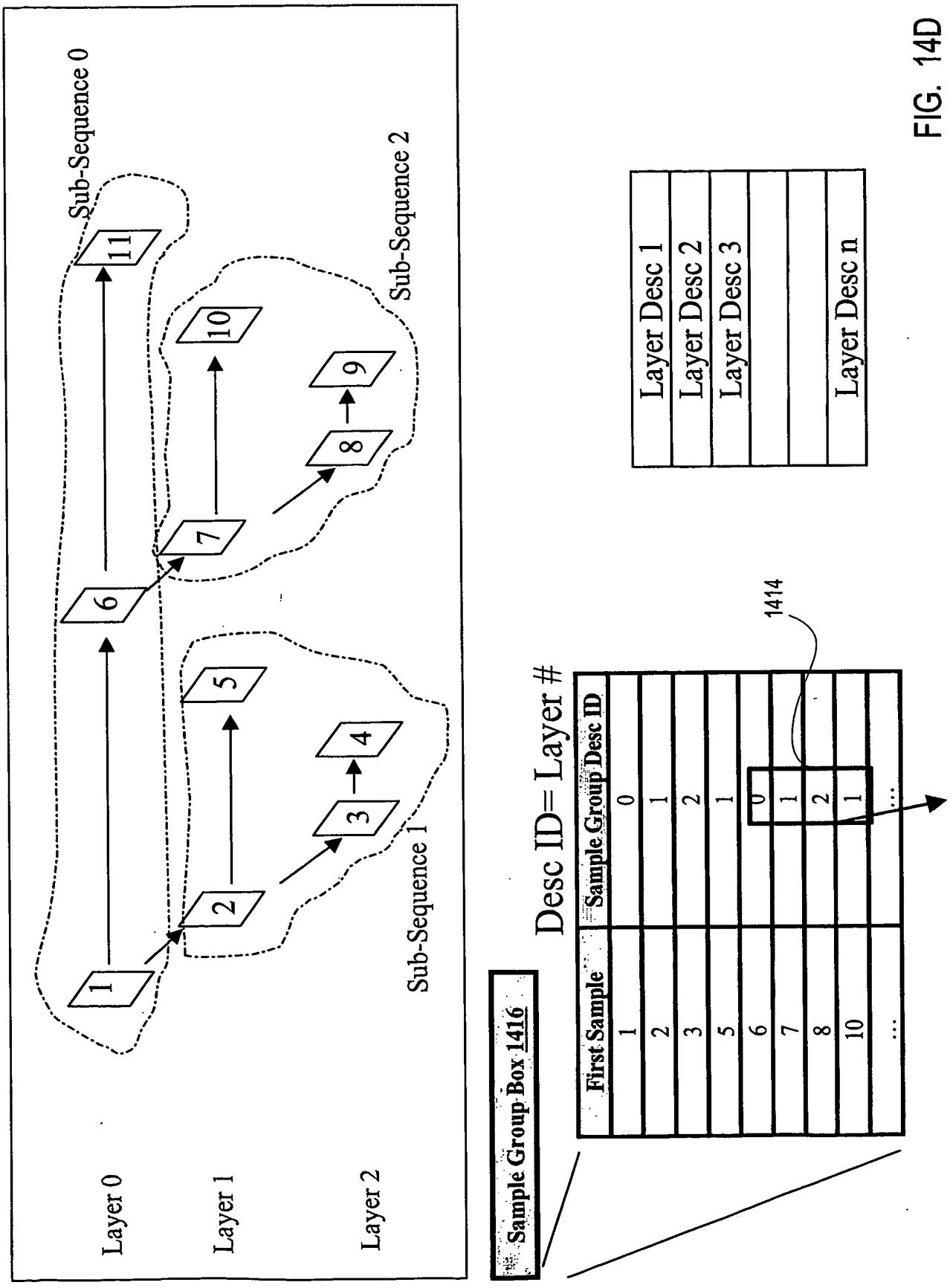
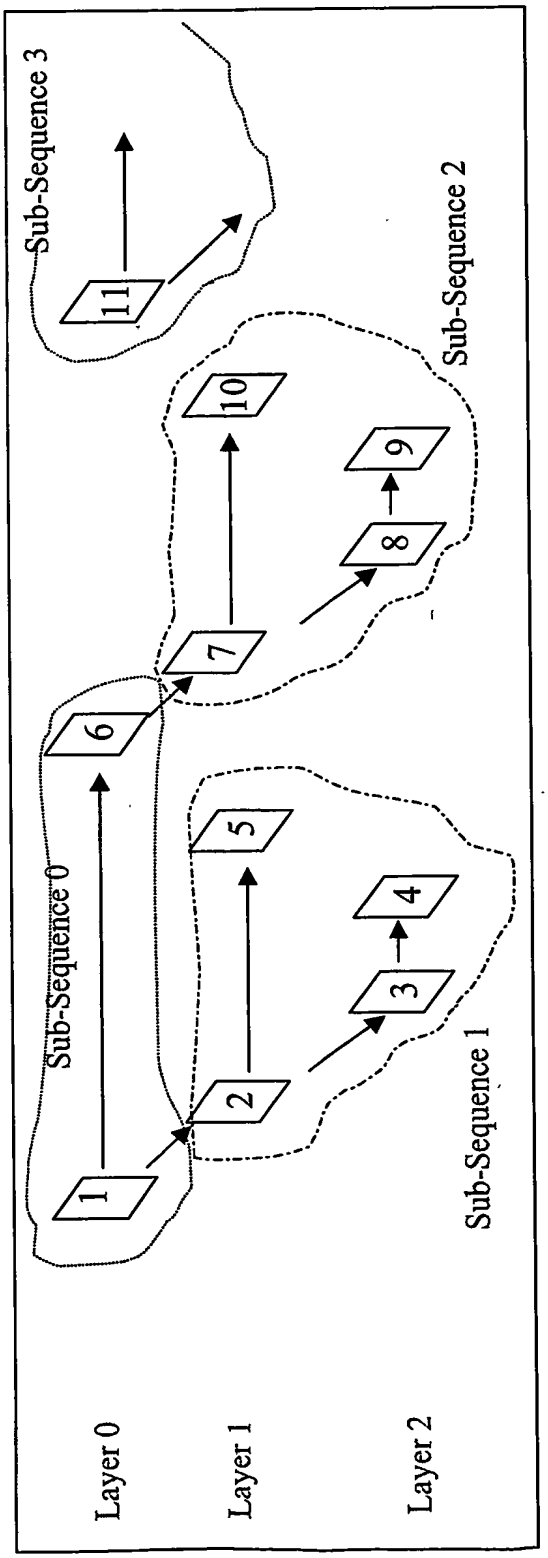

Sub-Sequence 0
Sub-Sequence 2
Sub-Sequence 1
Layer 0
Layer 1
Layer 2
1
2
3
4
5
6
7
8
9
10
11
Layer Desc 1
Layer Desc 2
Layer Desc 3
Layer Desc n
Sample Group Box 1416
Desc ID= Layer #
First Sample
Sample Group Desc ID
1414

FIG. 14D


Sub-Sequence 3
Sub-Sequence 2
Sub-Sequence 0
Sub-Sequence 1
Layer 0
Layer 1
Layer 2
1
2
3
4
5
6
7
8
9
10
11

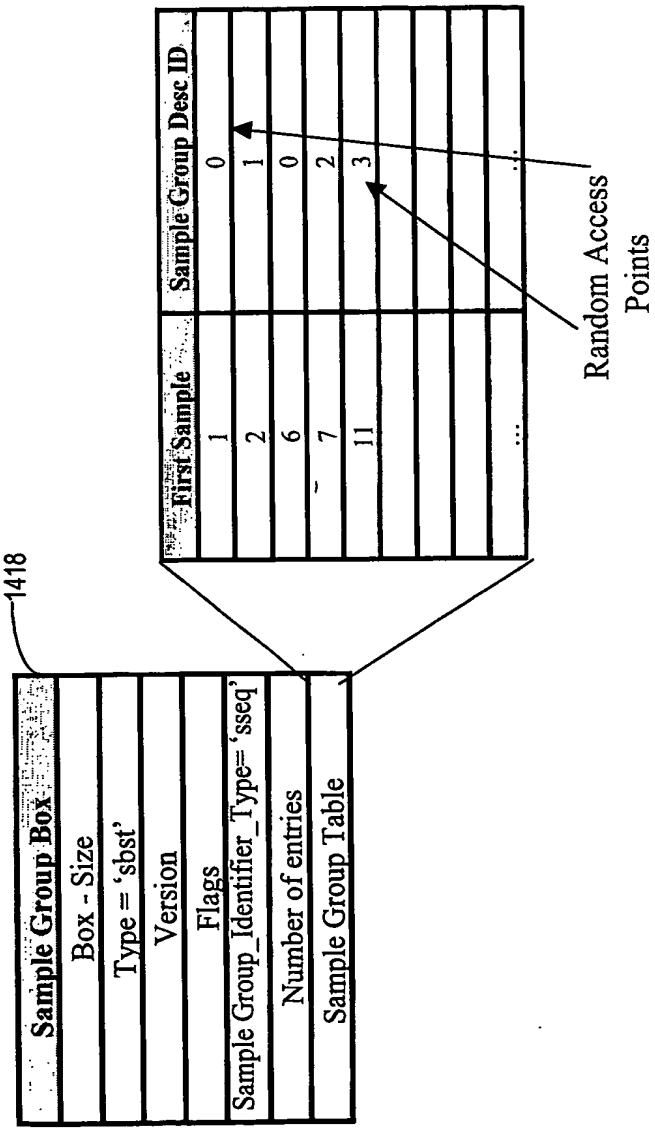

1418
Sample Group Box
Box - Size
Type = 'sbst'
Version
Flags
Sample Group_Identifier_Type= 'sseq'
Number of entries
Sample Group Table
First Sample
Sample Group Desc ID
1
0
2
1
6
0
7
2
11
3
...
...
Random Access Points

FIG. 14E

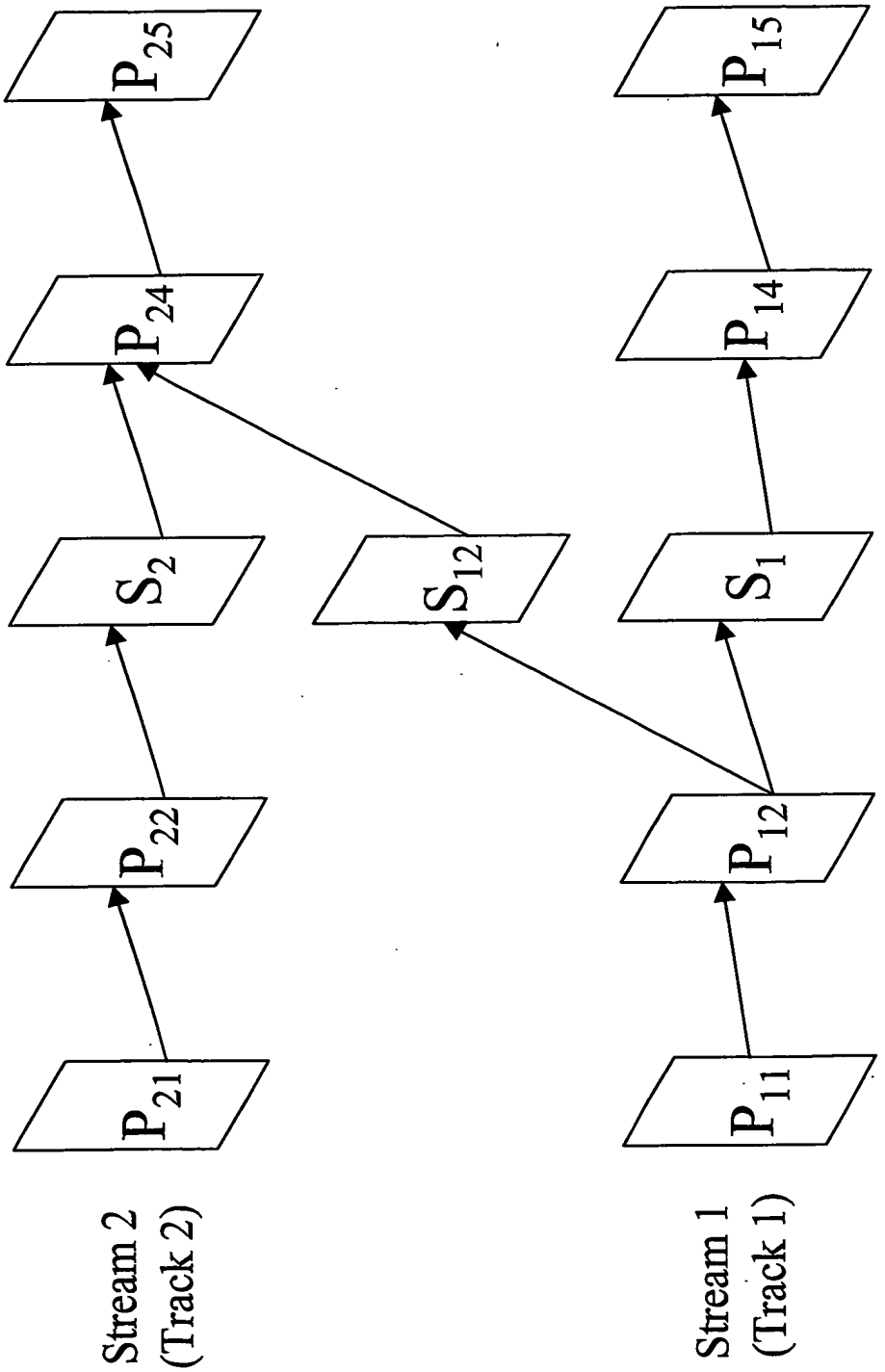

Stream 2
(Track 2)
Stream 1
(Track 1)
P21
P22
S2
P24
P25
S12
P11
P12
S1
P14
P15

FIG. 15A

Switch Sample Set 1502

Switch Sample	*Switch Sample Track*	*Reference Track*	*Reference Samples*
S_2	2	2	P_{22}
S_{12}	xxx	1	P_{12}

FIG. 15B

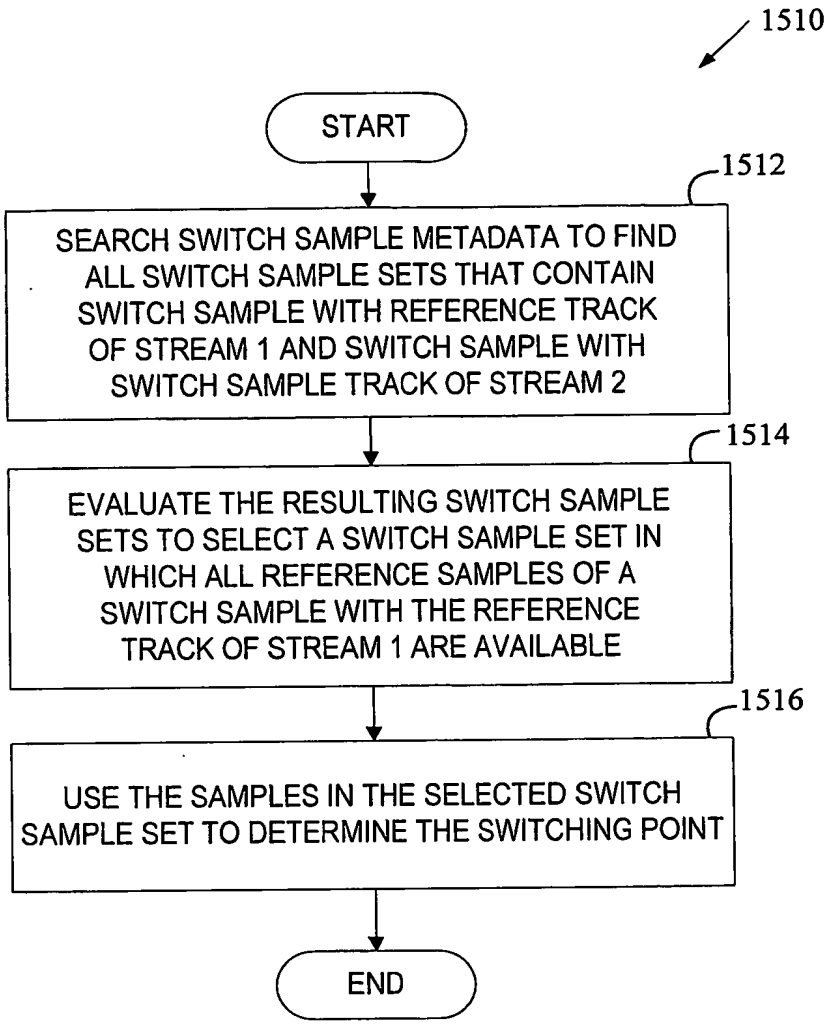

1510
START
1512
SEARCH SWITCH SAMPLE METADATA TO FIND ALL SWITCH SAMPLE SETS THAT CONTAIN SWITCH SAMPLE WITH REFERENCE TRACK OF STREAM 1 AND SWITCH SAMPLE WITH SWITCH SAMPLE TRACK OF STREAM 2
1514
EVALUATE THE RESULTING SWITCH SAMPLE SETS TO SELECT A SWITCH SAMPLE SET IN WHICH ALL REFERENCE SAMPLES OF A SWITCH SAMPLE WITH THE REFERENCE TRACK OF STREAM 1 ARE AVAILABLE
1516
USE THE SAMPLES IN THE SELECTED SWITCH SAMPLE SET TO DETERMINE THE SWITCHING POINT
END

FIG. 15C

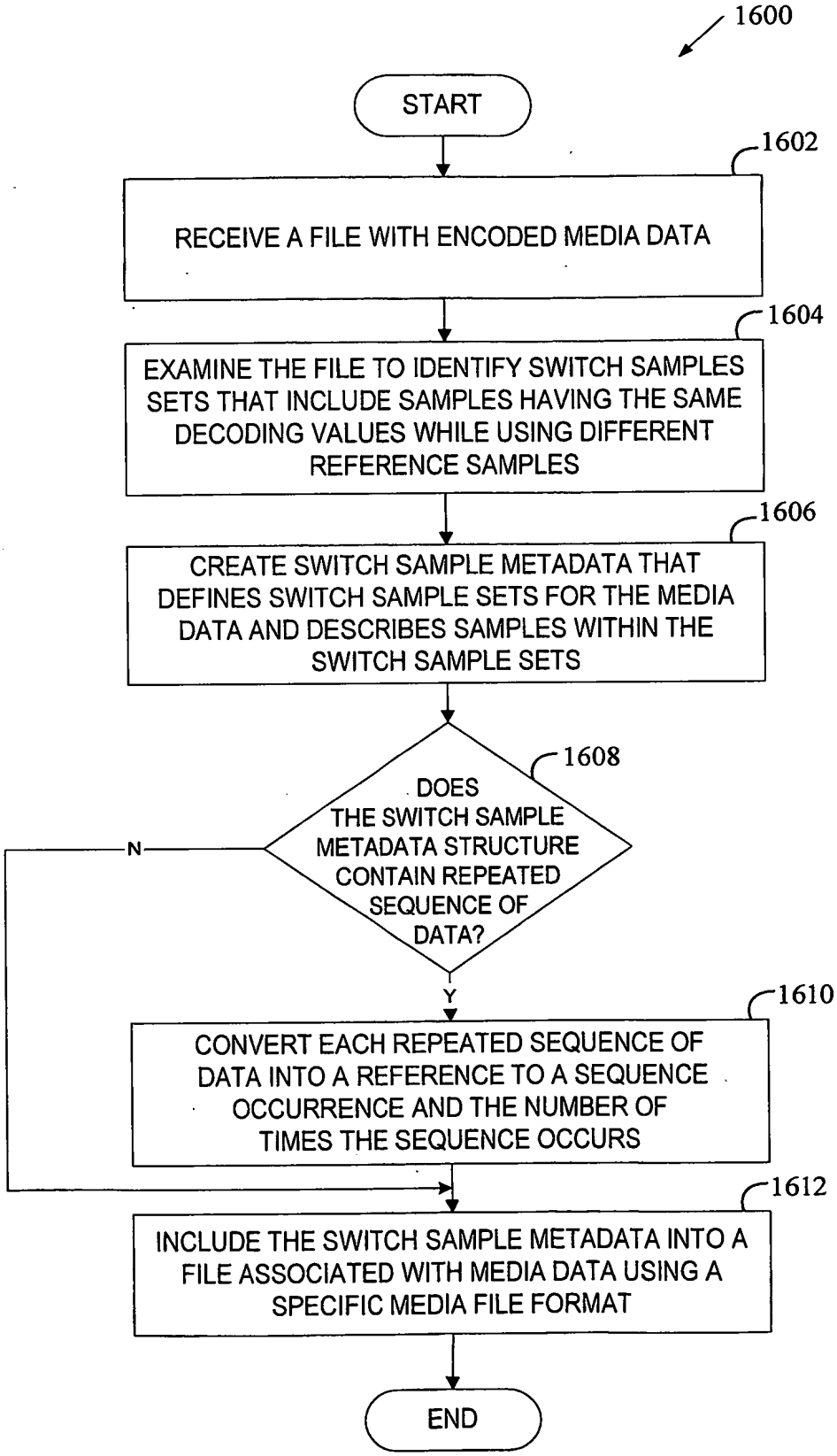

1600
START
1602
RECEIVE A FILE WITH ENCODED MEDIA DATA
1604
EXAMINE THE FILE TO IDENTIFY SWITCH SAMPLES SETS THAT INCLUDE SAMPLES HAVING THE SAME DECODING VALUES WHILE USING DIFFERENT REFERENCE SAMPLES
1606
CREATE SWITCH SAMPLE METADATA THAT DEFINES SWITCH SAMPLE SETS FOR THE MEDIA DATA AND DESCRIBES SAMPLES WITHIN THE SWITCH SAMPLE SETS
1608
DOES THE SWITCH SAMPLE METADATA STRUCTURE CONTAIN REPEATED SEQUENCE OF DATA?
N
Y
1610
CONVERT EACH REPEATED SEQUENCE OF DATA INTO A REFERENCE TO A SEQUENCE OCCURRENCE AND THE NUMBER OF TIMES THE SEQUENCE OCCURS
1612
INCLUDE THE SWITCH SAMPLE METADATA INTO A FILE ASSOCIATED WITH MEDIA DATA USING A SPECIFIC MEDIA FILE FORMAT
END

FIG. 16

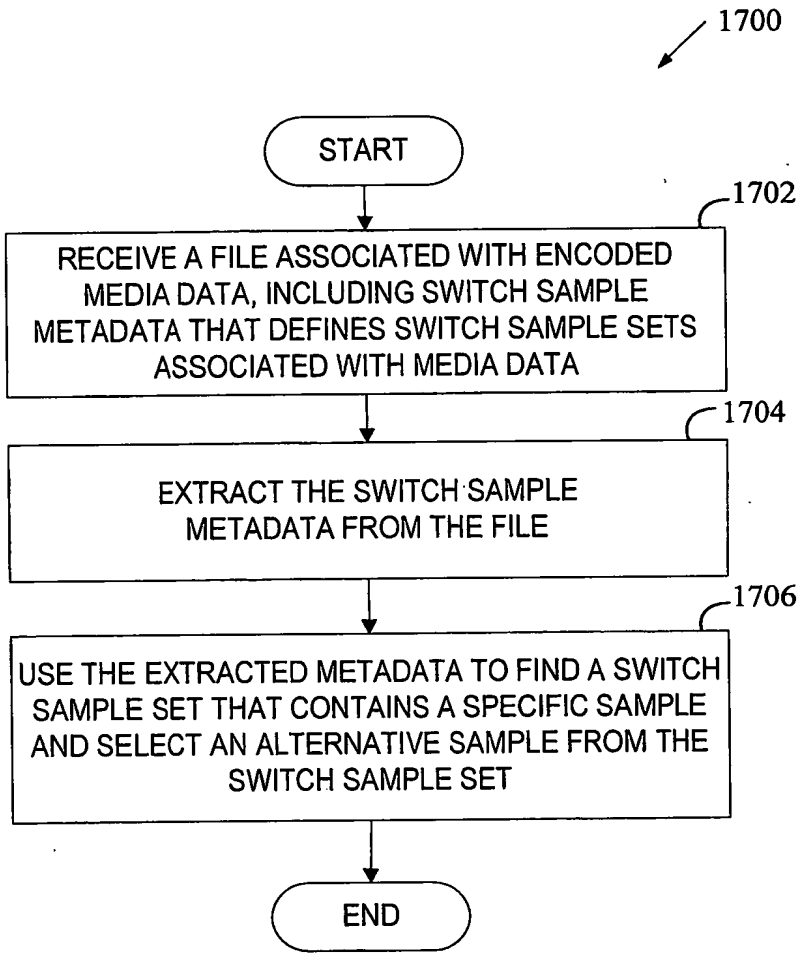

1700
START
1702
RECEIVE A FILE ASSOCIATED WITH ENCODED MEDIA DATA, INCLUDING SWITCH SAMPLE METADATA THAT DEFINES SWITCH SAMPLE SETS ASSOCIATED WITH MEDIA DATA
1704
EXTRACT THE SWITCH SAMPLE METADATA FROM THE FILE
1706
USE THE EXTRACTED METADATA TO FIND A SWITCH SAMPLE SET THAT CONTAINS A SPECIFIC SAMPLE AND SELECT AN ALTERNATIVE SAMPLE FROM THE SWITCH SAMPLE SET
END

FIG. 17

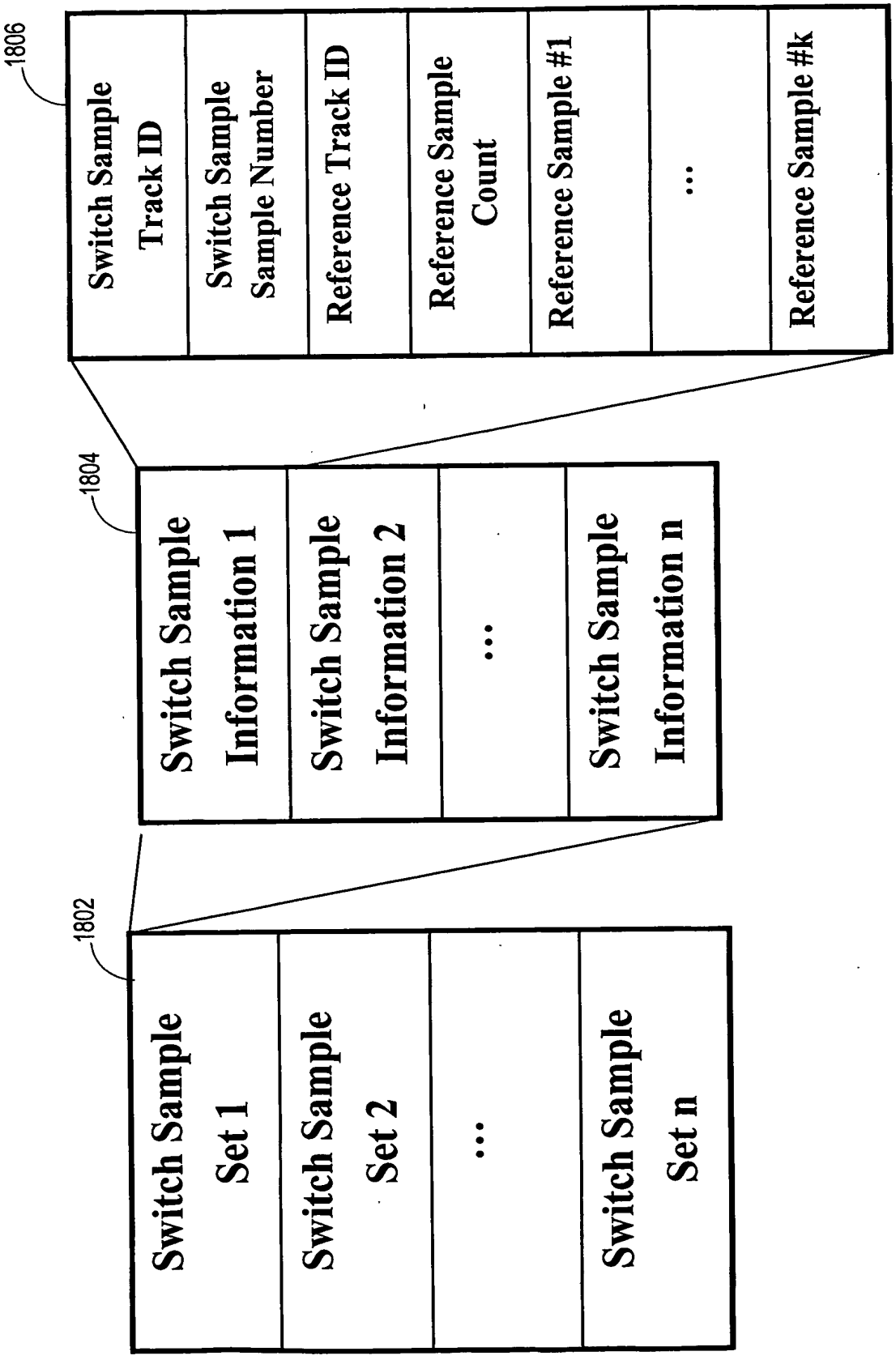

1802
Switch Sample Set 1
Switch Sample Set 2
...
Switch Sample Set n
1804
Switch Sample Information 1
Switch Sample Information 2
...
Switch Sample Information n
1806
Switch Sample Track ID
Switch Sample Sample Number
Reference Track ID
Reference Sample Count
Reference Sample #1
...
Reference Sample #k

FIG. 18

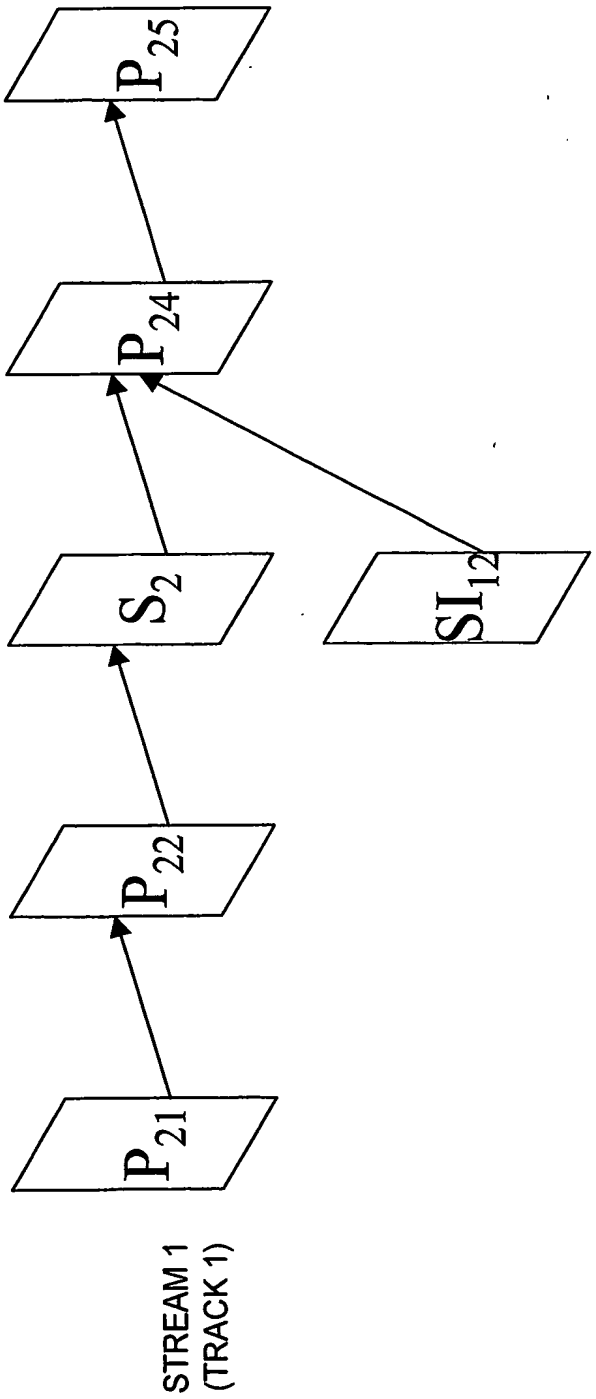


FIG. 19A

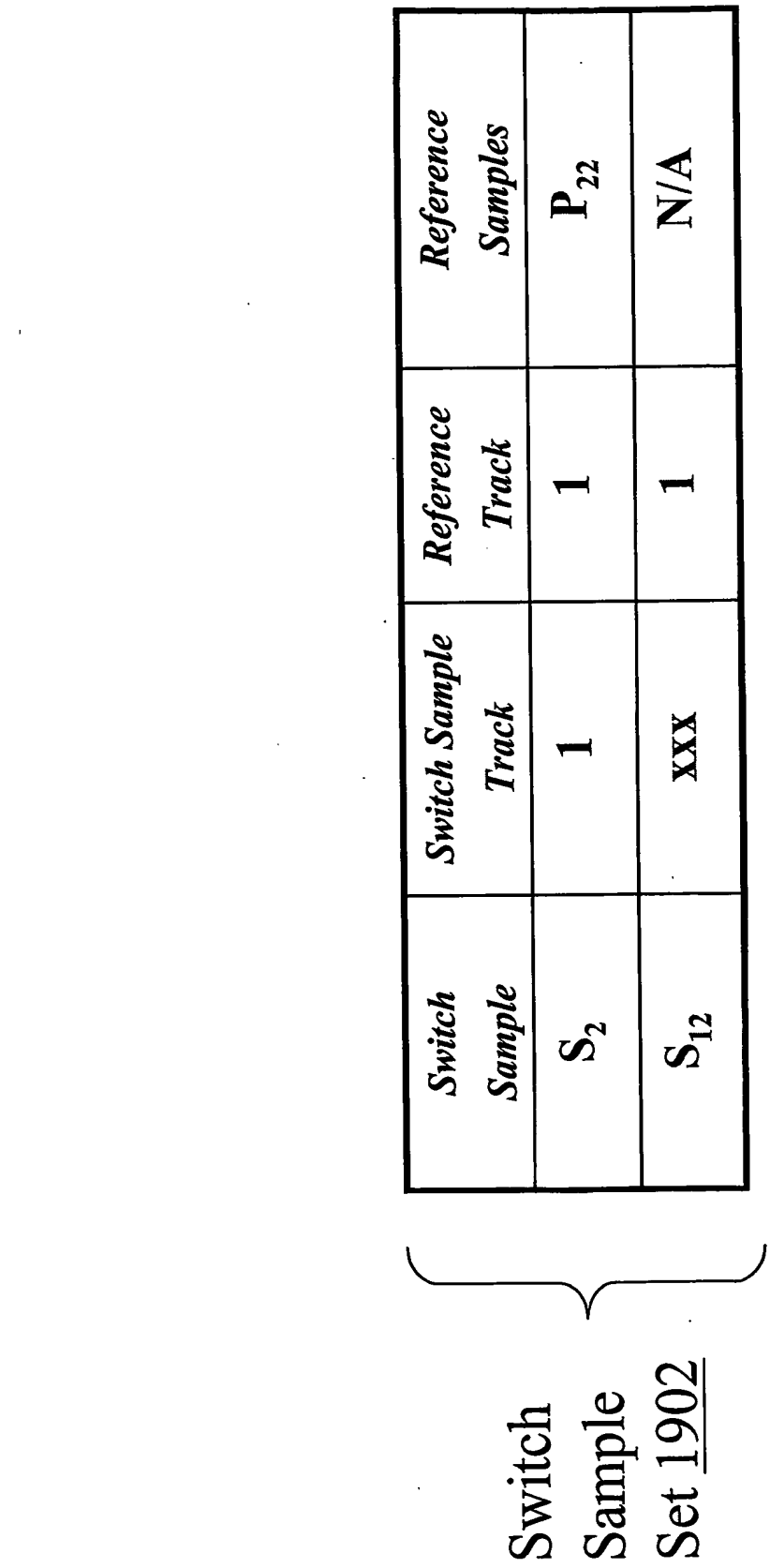

Switch Sample Set 1902
Switch Sample
Switch Sample Track
Reference Track
Reference Samples
S2
1
1
P22
S12
XXX
1
N/A

FIG. 19B

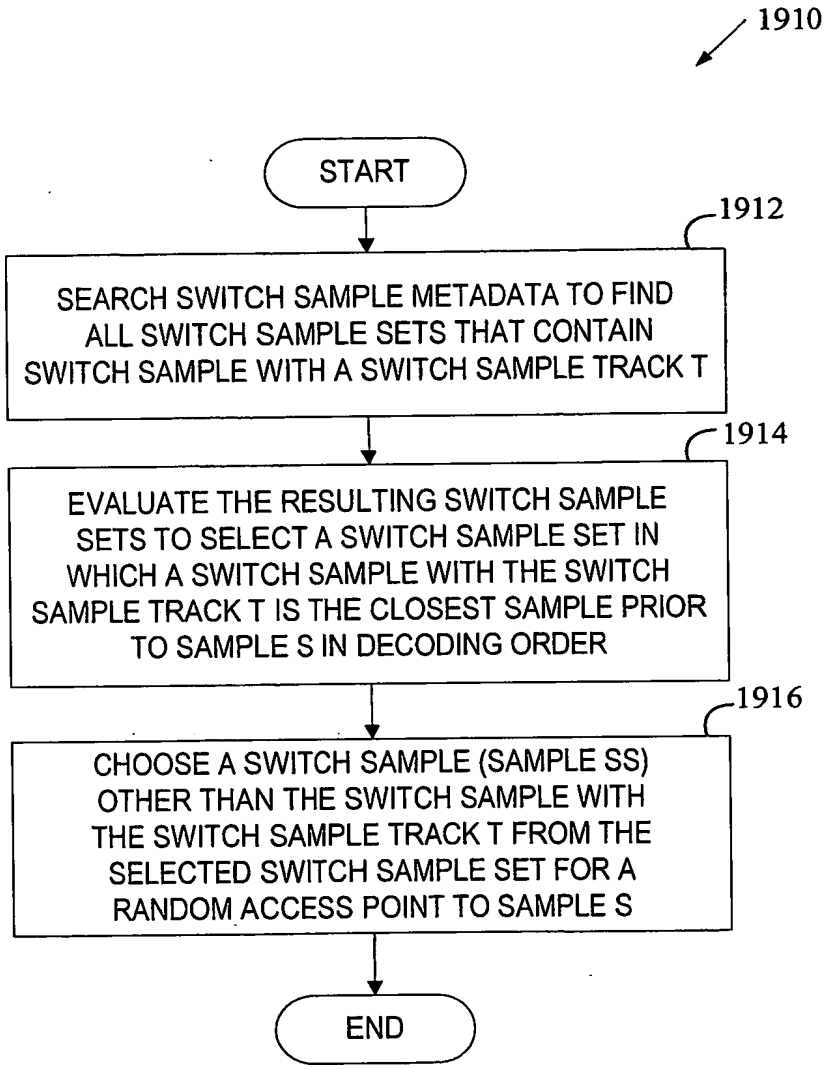

1910
START
1912
SEARCH SWITCH SAMPLE METADATA TO FIND ALL SWITCH SAMPLE SETS THAT CONTAIN SWITCH SAMPLE WITH A SWITCH SAMPLE TRACK T
1914
EVALUATE THE RESULTING SWITCH SAMPLE SETS TO SELECT A SWITCH SAMPLE SET IN WHICH A SWITCH SAMPLE WITH THE SWITCH SAMPLE TRACK T IS THE CLOSEST SAMPLE PRIOR TO SAMPLE S IN DECODING ORDER
1916
CHOOSE A SWITCH SAMPLE (SAMPLE SS) OTHER THAN THE SWITCH SAMPLE WITH THE SWITCH SAMPLE TRACK T FROM THE SELECTED SWITCH SAMPLE SET FOR A RANDOM ACCESS POINT TO SAMPLE S
END

FIG. 19C

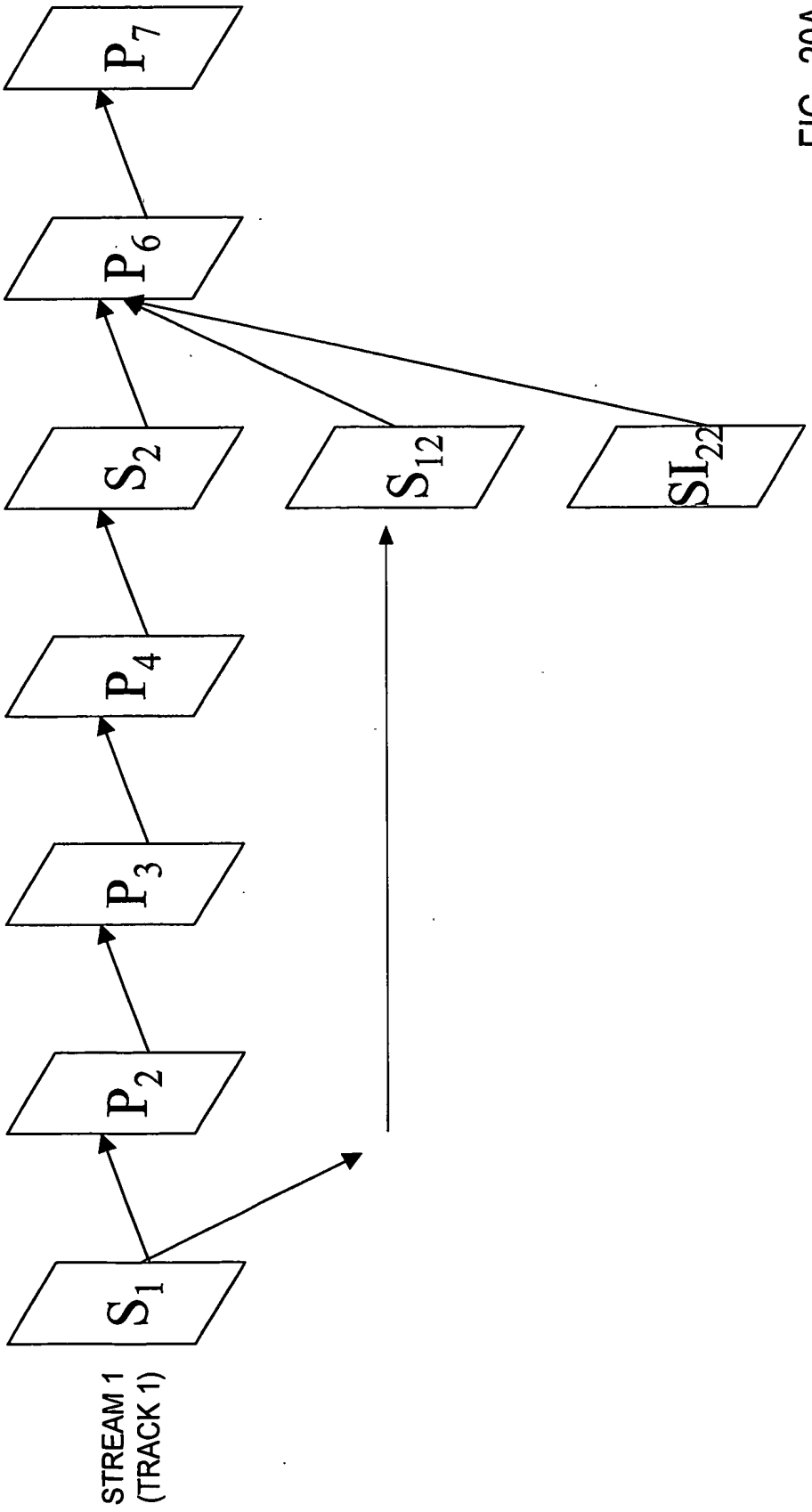
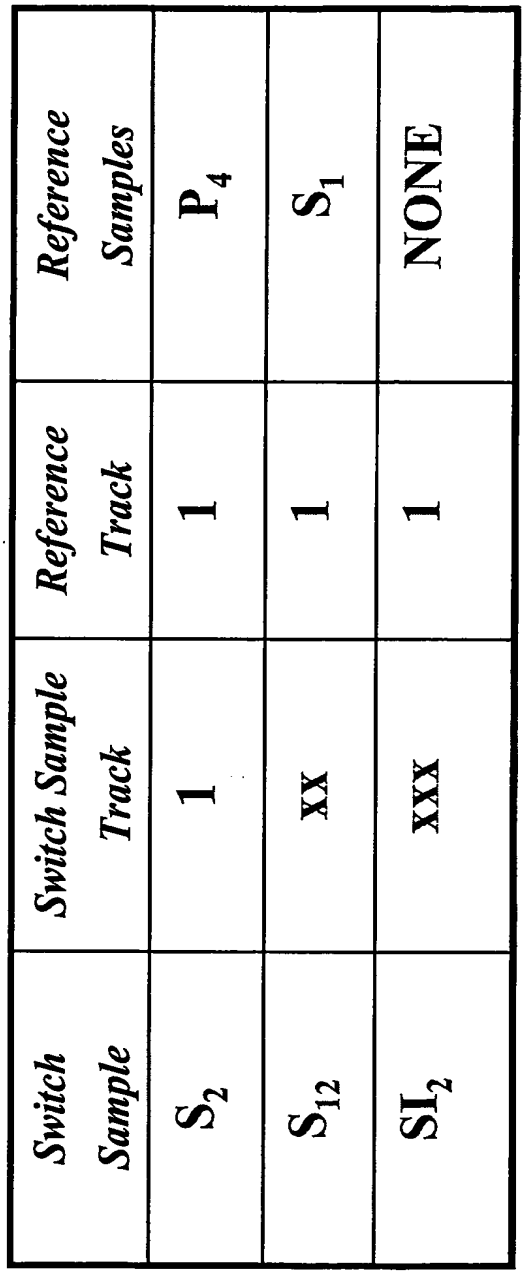


FIG. 20A



Switch Sample Set 2002

Switch Sample	*Switch Sample Track*	*Reference Track*	*Reference Samples*
S_2	1	1	P_4
S_{12}	XX	1	S_1
SL_2	XXX	1	NONE

FIG. 20B

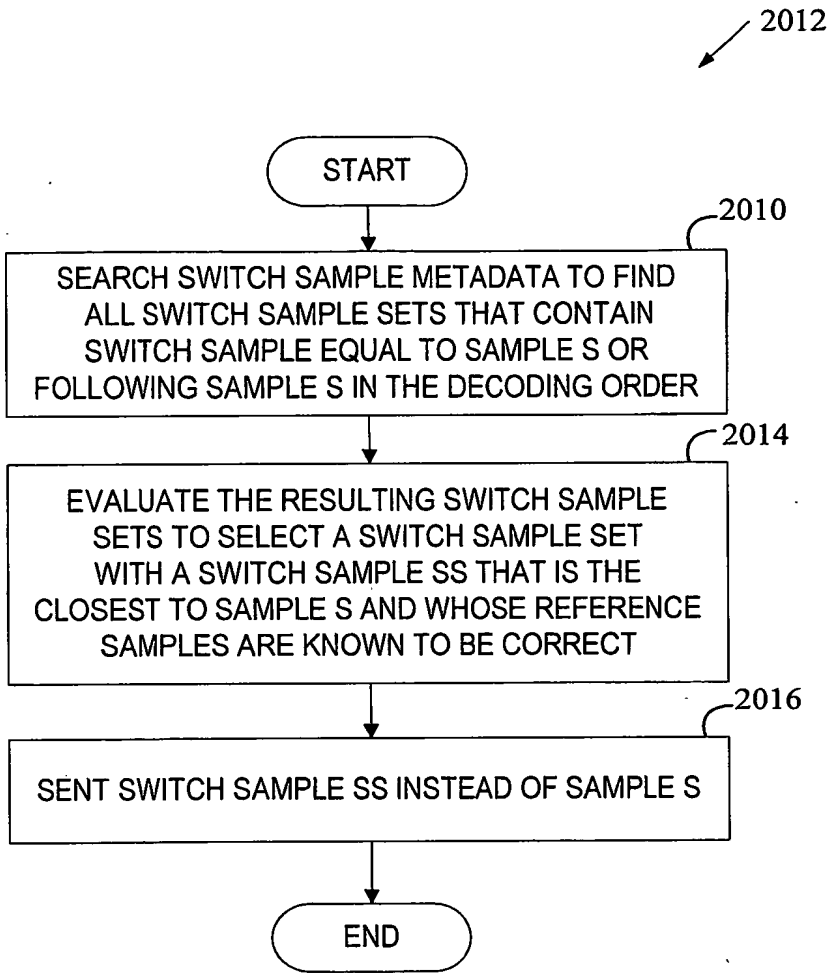

2012
START
2010
SEARCH SWITCH SAMPLE METADATA TO FIND ALL SWITCH SAMPLE SETS THAT CONTAIN SWITCH SAMPLE EQUAL TO SAMPLE S OR FOLLOWING SAMPLE S IN THE DECODING ORDER
2014
EVALUATE THE RESULTING SWITCH SAMPLE SETS TO SELECT A SWITCH SAMPLE SET WITH A SWITCH SAMPLE SS THAT IS THE CLOSEST TO SAMPLE S AND WHOSE REFERENCE SAMPLES ARE KNOWN TO BE CORRECT
2016
SENT SWITCH SAMPLE SS INSTEAD OF SAMPLE S
END

FIG. 20C

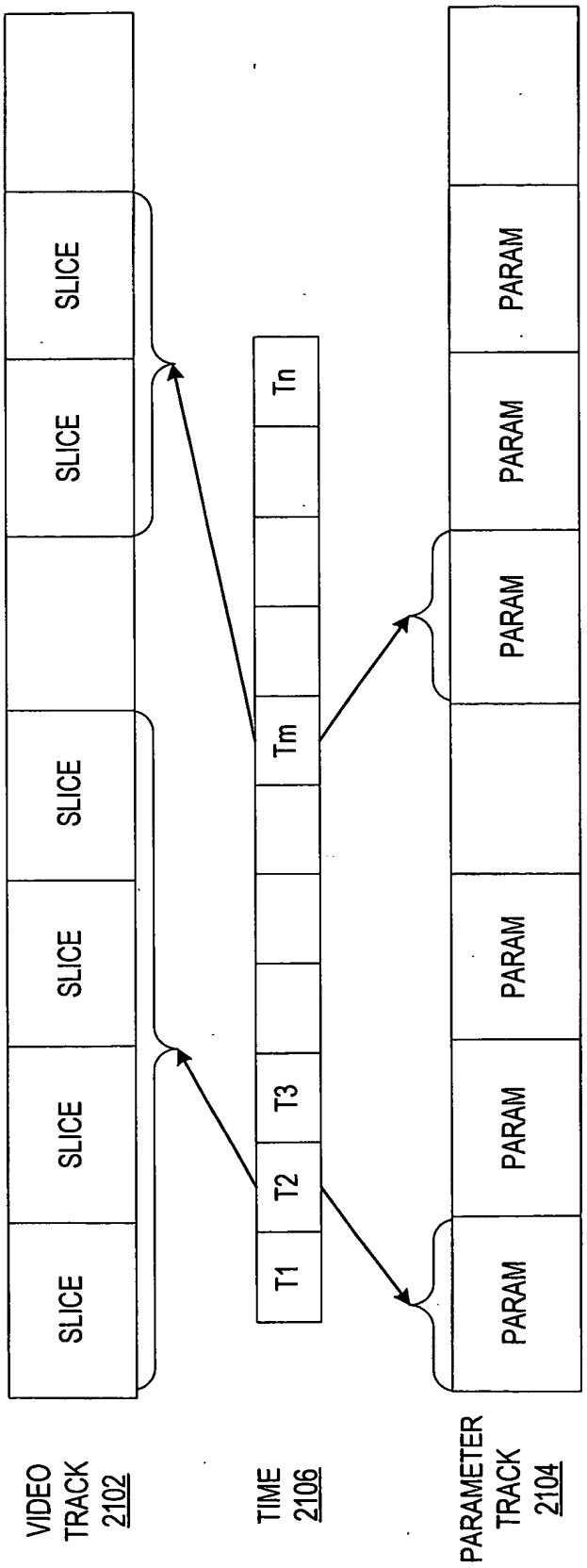

VIDEO SAMPLE
VIDEO SAMPLE
SLICE
SLICE
SLICE
SLICE
SLICE
SLICE
SLICE
SLICE
T1
T2
T3
Tm
Tn
PARAM
PARAM
PARAM
PARAM
PARAM
PARAM
PARAM
PARAM
VIDEO TRACK 2102
TIME 2106
PARAMETER TRACK 2104
PARAMETERSET SAMPLE

FIG. 21

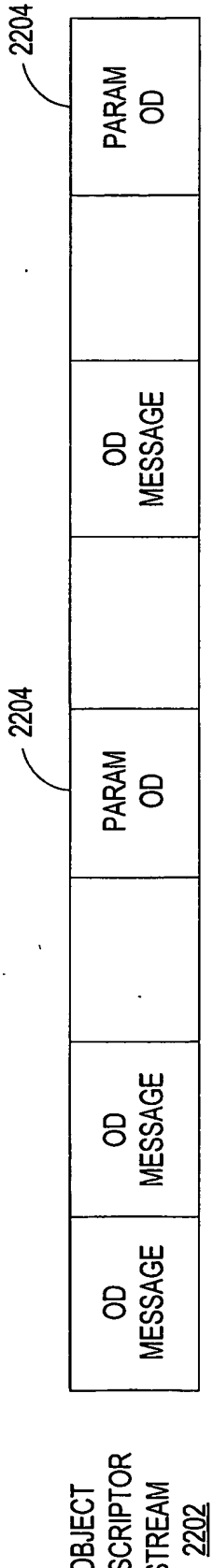

OBJECT DESCRIPTOR STREAM 2202
OD MESSAGE
OD MESSAGE
PARAM OD
2204
OD MESSAGE
PARAM OD
2204

FIG. 22

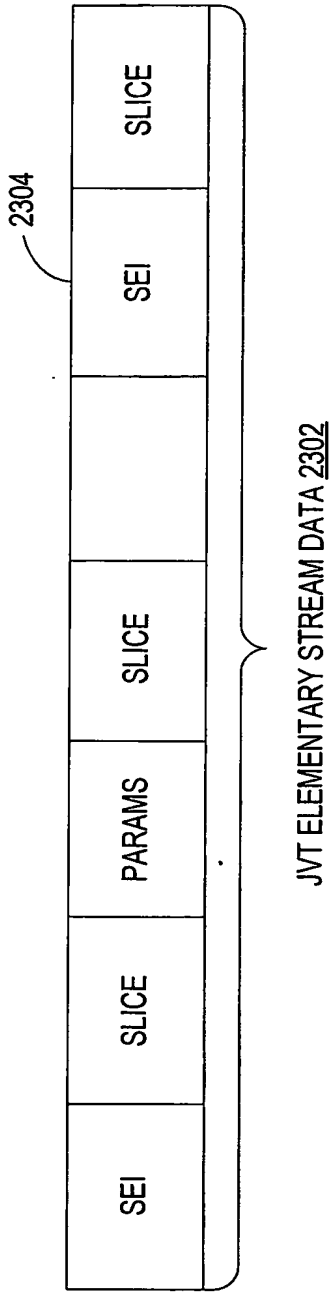

2304
SEI
SLICE
PARAMS
SLICE
SEI
SLICE
JVT ELEMENTARY STREAM DATA 2302

FIG. 23

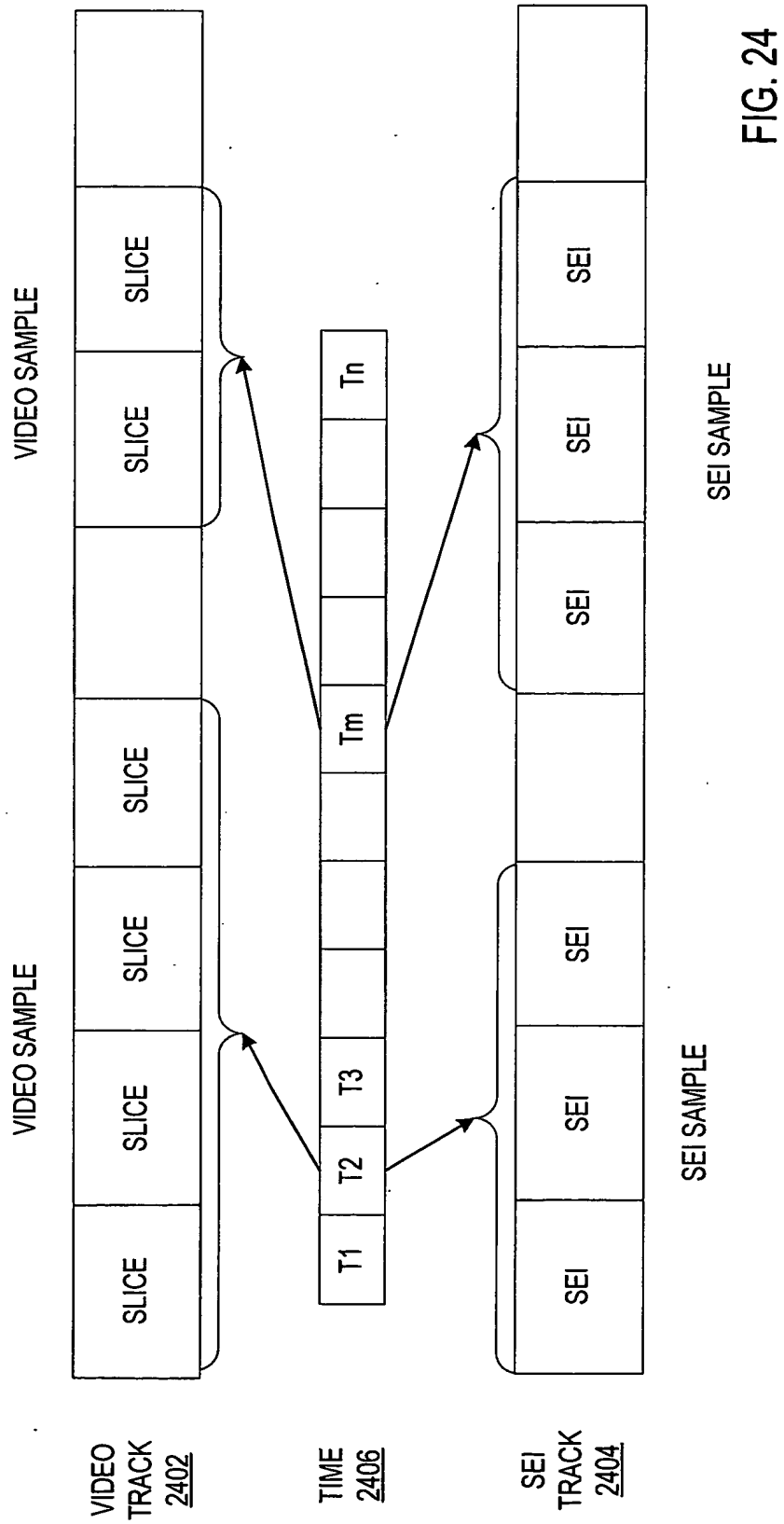

VIDEO TRACK 2402
TIME 2406
SEI TRACK 2404
VIDEO SAMPLE
VIDEO SAMPLE
SEI SAMPLE
SEI SAMPLE
SLICE
SEI
T1
T2
T3
Tm
Tn

FIG. 24

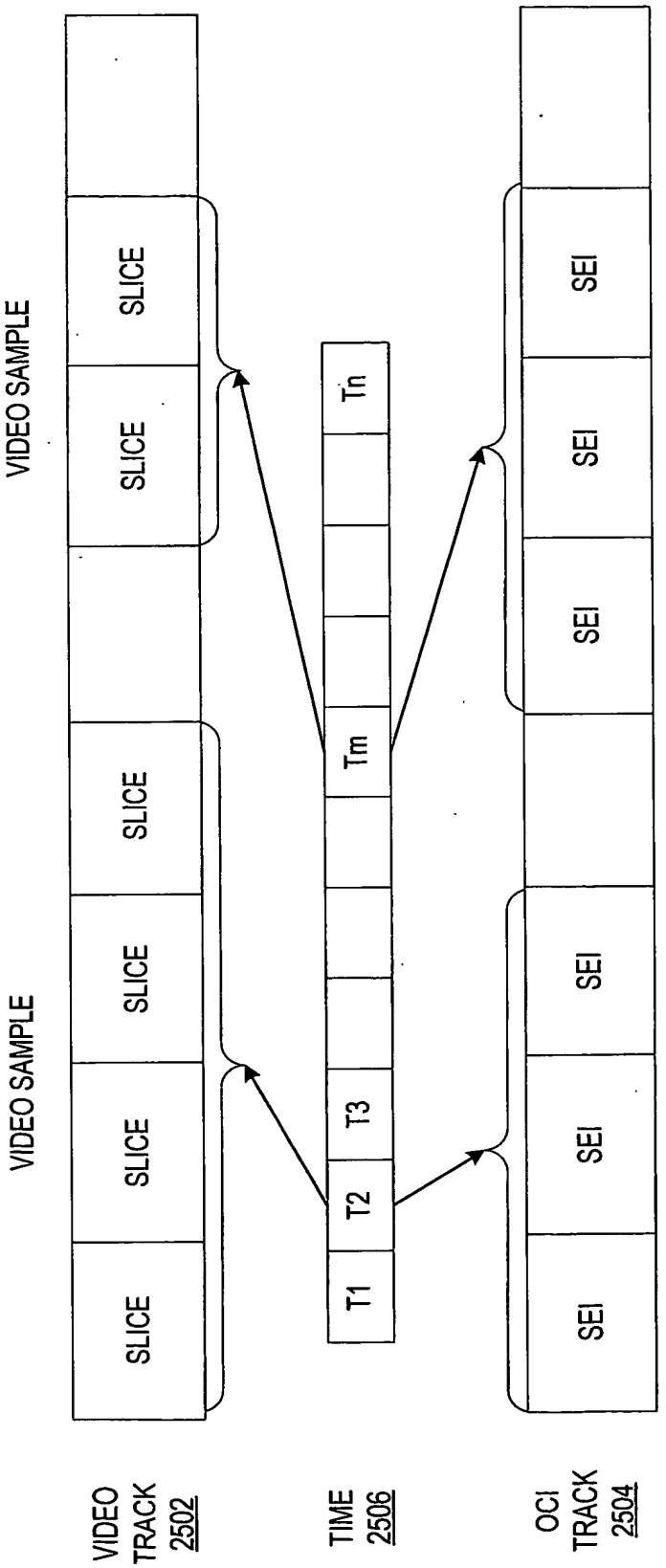

VIDEO SAMPLE
VIDEO SAMPLE
SLICE
SLICE
SLICE
SLICE
SLICE
SLICE
T1
T2
T3
Tm
Tn
SEI
SEI
SEI
SEI
SEI
SEI
SEI SAMPLE
SEI SAMPLE
VIDEO TRACK 2502
TIME 2506
OCI TRACK 2504

FIG. 25

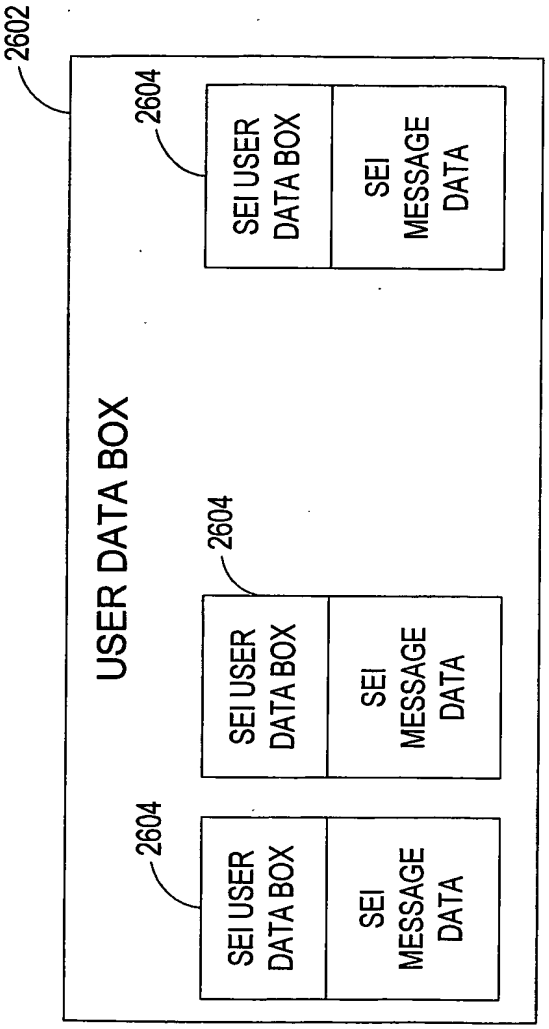

2602
USER DATA BOX
2604
SEI USER DATA BOX
SEI MESSAGE DATA
2604
SEI USER DATA BOX
SEI MESSAGE DATA
2604
SEI USER DATA BOX
SEI MESSAGE DATA

FIG. 26

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US03/13145

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 17/30
US CL : 707/1

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 707/1, 348/461, 709/230, 382/305, 700/17

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X,P	US 6,400,996 B1 (HOFFBERG et al) 04 June 2002 (04.06.2002), Abstract; column 50, line 53 through column 52, line 27; column 93, lines 8-23; column 98, lines 50-67; column 100,	1-3
---		---
Y,P	lines 52-65; column 104, lines 25-39; column 106, lines 1-64; column 121, line 66 through column 122, line 14; column 131, lines 25-45; column 149, lines 34-61.	4-34
Y,P	US 6,426,778 B1 (VALDEZ, JR) 30 July 2002 (30.07.2002), Abstract; column 3, line 40 through column 4, line 32; column 10, lines 9-27; column 16, line 8 through column 17, line 19; column 19, line 45 through column 20, line 64; and column 21, line 26 through column 22, line 48).	4-11, 27-34
Y	US 5,832,472 (SHEPPARD, II) 03 November 1998 (03.11.1998), Abstract; column 3, line 29 through column 4, line 18; column 6, lines 32-53; column 7, lines 3-30; and column 12, lines 26-66).	12-26, 27-34
Y,P	US 6,453,355 B1 (JONES et al) 17 September 2002 (17.09.2002), Abstract; column 1, lines 9-13; and column 12, line 61 through column 13, line 20.	19-20
Y, E	US 6,574,378 B1 (LIM) 03 June 2003 (03.06.2003), Abstract; column 3, line 63 through column 4, line 43; column 8, lines 39-67; and column 11, lines 37-56.	24-26

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
09 June 2003 (09.06.2003)

Date of mailing of the international search report
14 JUL 2003

Name and mailing address of the ISA/US
Mail Stop PCT, Attn: ISA/US
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
Facsimile No.

Authorized officer
Tony Mahmoudi

Telephone No. 703-305-4887

Form PCT/ISA/210 (second sheet) (July 1998)